LEVEL II

RADC-TR-81-255
Final Technical Report
September 1981

MULTI-SENSOR SCENE SYNTHESIS AND ANALYSIS

Digital Decision Systems, Inc.

Ernest L. Hall
Rafael C. Gonzalez

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

82

RADC-TR-81-255 has been reviewed and is approved for publication.

APPROVED: Robert H. La Salle
ROBERT H. LASALLE
Project Engineer

APPROVED: [illegible]
JOHN N. ENTZMINGER
Technical Director
Intelligence and Reconnaissance Division

FOR THE COMMANDER: [illegible]
JOHN P. HUSS
Acting Chief, Plans Office

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-81-255	2. GOVT ACCESSION NO. AD-A110 812	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) MULTI-SENSOR SCENE SYNTHESIS AND ANALYSIS		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 1 Feb 80 to 30 Apr 81
		6. PERFORMING ORG. REPORT NUMBER N/A
7. AUTHOR(s) Ernest L. Hall Rafael C. Gonzalez		8. CONTRACT OR GRANT NUMBER(s) F30602-80-C-0084
9. PERFORMING ORGANIZATION NAME AND ADDRESS Digital Decision Systems, Inc. P. O. Box 8315, University Station Knoxville TN 37916		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 45941721
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (IRRP) Griffiss AFB NY 13441		12. REPORT DATE September 1981
		13. NUMBER OF PAGES 324
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Robert H. LaSalle (IRRP)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Image Assembly System	Scene Representation
Image Understanding	Hidden Surface Algorithms
Scene Synthesis	Surface Shading
Segmentation	Texture Processing

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes research work performed by Digital Decision Systems, Inc. on the Multi-Sensor Scene Synthesis and Analysis Study under Contract Number F30602-80-C-0084 for the Rome Air Development Center during the period of 1 February 1980 to 30 April 1981.

This technical report contains a detailed analysis of digital computer image analysis and scene synthesis algorithms and techniques in order to

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

provide the basis for the design of an Image Assembly System capable of functioning as a flexible research aid for both areas.

A survey of common image and scene analysis algorithms is presented which includes image enhancement techniques as well as modern scene segmentation and object location techniques. A special emphasis is ~~also~~ given to image representation models using relational tables and data structures for image analysis.

A survey of important scene synthesis algorithms is ~~also~~ given which includes three dimensional scene representation, hidden surface elimination, and surface shading techniques.

The requirements in terms of hardware and software for separate analysis and synthesis systems are ~~then~~ presented. Using the previous survey of algorithms and system requirements as a basis, a performance specification for the Image Assembly is defined. An example system consisting of available hardware elements is presented to demonstrate the practicality of the performance specification. Key developments required in special purpose hardware, software, and algorithms are presented as recommendations for further work.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Objective

The Multi-Sensor Scene Synthesis and Analysis Study was conducted by Digital Decision Systems, Inc., under contract F30602-80-C-0084 for the Rome Air Development Center during the period of 1 February 1980 to 1 April 1981. The fundamental problem considered in this research was the design of a computer imaging system which contained the combined capabilities of modern image analysis and scene synthesis. These combined capabilities, if made available in a compact, easy-to-use system, would provide an unparalleled research facility for image and scene analysis and synthesis.

The objective of this study was to investigate modern image analysis and scene synthesis techniques in relation to key aspects of a computer image generation, storage, and usage system. Through the application of a variety of image processing algorithms and control software, a flexible image analysis and synthesis research aid could be established. This aid, called the Image Assembly System (IAS), is described in this report. Special emphasis has been given to image construction flexibility and processing system design requirements. The principal result of this effort is the design of a versatile facility for investigating fundamental aspects of image usage.

The significance of this work is related to the increasing amount of digital image analysis and scene synthesis being applied to advanced Air Force applications. Typical applications range from the generation of airborne sensor image predictions, flight simulator imaging systems, terminal guidance, image processing and reference scene generation, and image compression and reconstruction. Each of these applications entails computer processing and storage of image and scene information. Because of the large amount of information in the data and the requirement of real time processing, the design factors such as speed, cost, size, and complexity must be carefully assessed.

The Image Assembly System design presented in this report was arrived at after a thorough literature search in both the image analysis and scene synthesis areas, a study of the algorithms used for image analysis and scene synthesis, and a study of currently available hardware and software systems. The underlying hypothesis has been the investigation of the feasibility of combining both analysis and synthesis features into a single system to provide a powerful research aid.

A summary of the principal areas of research and development in scene analysis is presented in Section 1.2, and a corresponding summary for scene synthesis is given in Section 1.3. An outline of this report is given in Section 1.4.

1.2 Scene Analysis Summary

The purpose of digital scene analysis procedures is to arrive at an understanding of the elements composing a scene and their relationships. The definition of "understanding" for a specific application depends on the nature of the problem one wishes to solve. In analyzing a scene of military interest, for example, the desired level of understanding may range from the relatively simple problem of determining if the scene contains any changes from a previously-observed scene, to the considerably more complex task of identifying specific targets and ranking them in order of importance.

State-of-the-art algortihms for scene analysis may be divided into four broad categories:

1) Sensing and preprocessing;

2) Segmentation;

3) Recognition and interpretation; and

4) Organization and processing of scene data.

1.2.1 Sensing and Preprocessing

The sensing problem is one of converting a physical scene into a form suitable for computer processing. Sensors are categorized by their response in the electromagnetic energy spectrum (e.g., x-ray, optical, infrared, microwave, and ultrasonic). Often, the choice of an imaging sensor is determined by the environment in which it is expected to operate. For example, under-water applications usually preclude the use of sensors other than low frequency devices. In situations where more than one type of sensor could be used, additional constraints such as resolution, size, weight, and cost play a deciding factor.

Although scene analysis is basically a three-dimensional problem, much of the present work in this area is carried out using planar (image) views. This is due both to limitations in three-dimensional sensor technology and to a lack of procedures for segmentation, recognition, and interpretation of three-dimensional data. Spatial relationships of objects in a scene are approximated by using approaches such as stereo image processing and range imaging.

Preprocessing techniques typically used in scene analysis include noise reduction, enhancement, and restoration. In applications requiring fast processing, the usual approach is to approximate a preprocessing function by a small template (e.g., a 9 x 9 template) and use spatial convolution implemented in special-purpose hardware.

1.2.2 Segmentation

The purpose of segmentation is to partition the scene space into meaningful regions. Segmentation techniques can be divided into two principal categories: point dependent and region dependent. Point dependent techniques deal with methods that examine the scene on a point-by-point basis. Examples in this area include intensity, color, and range thresholding. This type of processing is suitable for cases where the quantity being thresholded exhibits a small number of distinct variations (e.g., in segmenting bright objects from a dark, uniform background).

Region dependent techniques are based on regional properties of a scene. The most notable approaches in this area include template matching (e.g., gradient, line, and edge detectors), texture segmentation, edge segmentation, and region growing.

1.2.3 Recognition and Interpretation

Recognition is basically a labeling process; that is, the function of recognition algorithms is to identify each segmented object in a scene and to assign a label (e.g., road, vehicle, building) to that object. The design of recognition procedures for scene analysis consists of two basic steps: selection of a set of features or descriptors, and selection of a classification strategy.

The principal descriptors used in scene analysis are based on shape and amplitude (e.g., intensity) information. Shape descriptors attempt to capture invariant geometrical properties of an object. This has been an illusive goal which has lead to an impressive number of proposed techniques for shape description. Techniques for shape analysis and description are either global (region) or boundary oriented. Global techniques include principal axes analysis, texture, two- and three-dimensional moment invariants, geometrical descriptors such as perimeter squared/area and the extrema of a region, topological properties such as the Euler number, and decomposition into primary convex subsets. Boundary techniques are based on thinned or skeletonized scene components. They include: Fourier descriptors, chain codes, graph representations (of which strings and trees are special cases), and shape numbers. Variations and inconsistencies in shape are handled by the use of models, rubber masks, relaxation, and syntactic techniques.

Classification strategies in use today may be subdivided into two principal categories: decision-theoretic and structural. Decision-theoretic techniques are based on the use of decision (discriminant) functions. Given M pattern vector classes, $\omega_1, \omega_2, \ldots, \omega_M$, the decision-theoretic approach consists of identifying M decision functions $d_1(\underline{x})$, $d_2(\underline{x}), \ldots d_M(\underline{x})$ with the property that, for any pattern vector $\underline{x}^*$ from class ω_i, $d_i(\underline{x}^*) > d_j(\underline{x}^*)$, $j = 1,2,\ldots M$ $j \neq i$. The objective is to find

M decision functions such that this condition holds for all classes with minimum error in misclassification.

Structural methods of pattern recognition are attempts to describe fundamental relationships among pattern primitives via discrete mathematical models. The most widely used method is syntactic pattern recognition in which concepts and results in formal language theory provide the structural descriptions. By contrast, decision theoretic approaches deal with patterns on a strictly quantitative basis, thus largely ignoring interrelationships among the primitives.

The existence of recognizable and finitely describable structure is essential for success in the syntactic approach. The lack of general inference techniques has meant that applications thus far have been largely confined to pictorial patterns characterized by shapes which the designer perceives and can describe via a grammar; for example, handwritten characters, chromosomes, fingerprints, particle collision photographs, speech and physiological waveforms, and general plane contour shapes.

One of the most significant recent extensions of syntactic techniques has been the explicit inclusion of semantic evaluations simultaneously with syntactic analysis by means of attributed grammars. A pattern primitive is defined by two components: a token or symbol from a finite alphabet, and an associated list of attributes consisting of logical, numerical, or vector values. Basically, the symbolic component denotes a class of primitives and the attributes give the feature values for a specific instance of a primitive in a pattern; these values are usually obtained by nonsyntactic pattern recognition methods, such as discriminant analysis.

Interpretation may be viewed as the process which, together with recognition, assigns attribute values to the primitives in a given representation. Although most present interpretation approaches for digital scene analysis are primarily heuristic or interactive techniques, promising formal approaches are emerging which attempt to unify the concepts of semantic/syntactic information in a scene. While these techniques are not yet fully developed, there are numerous specialized applications where even a limited degree of interpretation based on recognition of important features could have a significant impact. Examples include the categorization of aerial scenes as being either of military or nonmilitary interest, the rejection of faulty electronic components in an assembly line, adaptive control of robots by visual feedback, and autonomous target detection systems.

1.2.4 Organization and Processing of Scene Data

From the viewpoint of computer data storage and manipulation, a digitized scene is a data base. Even with large, fast computers, the designers of a computer-based scene processing system cannot neglect the design aspects of data bases and data structures, because a successful data base design not only improves the efficiency of the data storage and manipulation operations, but enhances the documentation and verification of all algorithms used to process the data. It can in fact be shown that for many complex algorithms, the organization of the input and output data is as critical to the effectiveness of the algorithm as is any other single factor.

In the case of scene analysis, use of a hierarchical data base as the overall organization of symbolic representations of scenes implies a fundamental hierarchy ranging from the lowest level scene primitives to the highest level labeled forms. Use of a relational data base to store the characteristics of each node in the hierarchy has been found effective for representing the kind of detailed information that must appear with each item in a scene hierarchy.

The concepts underlying relational and hierarchical data bases are the foundations for a logical, efficient mechanism for scene analysis algorithms.

1.3 Scene Synthesis Summary

The purpose of scene synthesis is to provide an image representation of a three or more dimensional scene. Although the largest market for scene synthesis systems has been in flight simulators, the advantages and cost effectiveness of scene simulation in almost all aspects of science and engineering are now being applied.

The following table of application areas, market values, and percentages was recently presented in an IEEE Tutorial on Computer-Assisted Design and Engineering.*

APPLICATIONS 1979

	M$	
Electronic CAD/CAM	400	27%
Mechanical CAD/CAM	350	24%
Cartography	210	14%
Process Control	115	8%
Business (MIS)	70	5%
Art and Animation	30	2%
Other	275	20%
TOTAL	1450	100%

*From Tutorial Notes on Computer-Assisted Design and Engineering, IEEE Spring COMPCON, San Francisco, CA, February, 1981.

Significant accomplishments can easily be cited for each area in the table. Perhaps the area of widest impact in terms of electronic applications is the design of VLSI integrated circuits. Not only the circuit elements and interconnections, but three dimensional electromagnetic circuit components as well, can be analyzed by computer and final artwork can be produced. Even though custom VLSI chip production is currently in the cost range of $40,000, the projected costs in the next few years may be as low as $500.

Exciting applications in mechanical design are also occurring. A popular three-dimensional surface display program, MOVIE.BYU, developed at Bringham Young University, is reported to have nearly 2,000 users throughout the world. This program permits polygonal representation of three-dimensional solids, as well as deformation and stress analysis. One of the most interesting cartographic applications is the Presidental Information System, which is a color graphics system designed to provide our national leaders with the latest demographic information, as well as current information in areas such as employment rates and productivity throughout the nation. Some of the most important process control applications are being conducted in the nuclear industry. For example, TVA produces a full, three-dimensional computer model of each of its nuclear reactor designs before it is built, then maintains the computer model throughout construction and operation.

Due to the high cost of scene simulation, most business applications are presently two-dimensional graphs and plots of important data. The art and animation areas of application are perhaps the best known.

A recent example is the computer scenes in the popular movie, Star Wars. Note that the other applications listed in the table contain a relatively large percentage--20%--of the market. These include the military applications as considered in this report, such as flight simulation, sensor simulation, and reference scene preparation for missile guidance.

Scene synthesis consists of the processes by which a real or synthetic scene can be input, represented, characterized, and transformed for viewing or analysis. This scene synthesis study was directed toward the definition of algorithms for interactive or automatic construction of both surface shapes and sensor response characteristics. The scene synthesis process was divided into four elements:

1) scene data generation;
2) surface description and representation;
3) surface characteristics; and
4) viewing transformations.

1.3.1 Scene Data Generation

Scene data generation consists of methods for generating the computer data base required for representing both topographic and man-made structures. Topographic data may consist of point and line data, surface data, and volume data. Man-made structures may consist of existing objects or synthetic objects. For existing objects, a variety of mensuration and photogrammetric techniques can be used for measurements of the surfaces. Also, several three-dimensional digitizers based upon mechanical, ultrasound, light, or imaging techniques are available. For synthetic objects, structures may be input by analytical specification, digitization from multiview drawing, building block techniques, or polygonal surface specification, each with interactive editing and constraint specification.

1.3.2 Surface Description and Representation

Surface description consists of constructing the internal computer data base representation of the scene. Several methods are available for representing both planar and curved surfaces. Patch representation requires the subdivision of a scene into regions bounded by a closed set of curves. The intersection vertices or knots are often the only information which the designer must specify with the form of the curve preselected. For example, a plane surface may be constructed between a set of three-dimensional vertices with a connection matrix specified by forming plane surfaces to fit the vertex array. To describe curved surfaces, several representations may be selected, including Bezier cubic polynomials, B-splines, Catmull cubic splines, or Coons surfaces. The data structures for surface description are also very important, since both the amount of storage required and the computation required for the viewing transformations are dependent upon the data structure. Several different data structures have been described in the literature, including a vertex list, vertex lists with pointers, pointers to edges, adjacent polygons, winged edge, curved surface canonical representations, and curved surface corner control point. A curve surface method will be required for the Image Assembly System. Also, since normal and tangent vectors will be required for display, these must be included in or easily computed from the data structure.

1.3.3 Surface Characteristics

Surface characteristics as viewed from any type sensor must be included in the Image Assembly System. In general, the elements which must be considered include the illumination source and shading mechanism, color characteristics, specular and highlight effects, texture generation, and shadow generation. Other special effects, such as wrinkled surface display, may also be important. The surface characteristics for a given sensor may be described in terms of objective characteristics such as illumination beam pattern, receiver point spread function, or nonlinear transfer function, gain and signal-to-noise ratio, and display format. To generate a particular display, real time viewing transformations are required.

1.3.4 Viewing Transformations

Viewing transformations, including the world space to eye space and the eye space to image space, may be implemented by a 4 x 4 matrix transformation. Also, clipping and windowing algorithms are available for real time display. Hidden surface elimination is the most difficult algorithm to implement in real time. Several methods, including the depth buffer algorithm, scan line algorithm, and the priority algorithms, were analyzed for possible inclusion in the Image Assembly System.

1.4 Outline of the Report

A detailed analysis of the algorithms for image analysis and scene synthesis are described in Sections 2 and 3. Included are the most commonly used as well as state-of-the-art techniques for image analysis such as image enhancement, segmentation, and feature extraction. Also included are key algorithms for scene synthesis such as discrete surface representation, hidden surface elimination, and surface shading.

The Image Assembly System design is described in Section 4. The hardware and software considerations are based largely upon the analysis of algorithms presented in Sections 2 and 3. General designs for both image analysis and scene synthesis are presented separately.

The performance specifications for the Image Assemb' System are presented in Section 5. This system has combined capabilities for performing the key algorithms and tasks for both image analysis and scene synthesis. Although general performance specifications are given, an example system of existing hardware components is also presented to demonstrate the practicality of the design.

Finally, the conclusions reached from this study and recommendations for further work are presented in Section 6.

2. SCENE ANALYSIS

2.1 Introduction

Digital scene analysis may be defined as the process of using a digital computer to extract, characterize, and interpret information from images of a three-dimensional world. Interest in scene analysis methods ranges from biomedical to industrial and military applications. Regardless of specific application, however, an essential characteristic of digital scene analysis techniques is the use of a machine for performing "intelligent" tasks, where the standard for intelligent behavior on the part of the machine is established by the capability of a human in successfully performing the same tasks.

The importance of scene analysis techniques in the broader field of machine intelligence can be easily grasped by a cursory review of the application areas in this field which are based on pictorial information. For example, a significant portion of current research in robotics as a means for enhancing industrial productivity is directed toward machines that, ideally, should be able to perceive and interpret information in a working environment such as an assembly area. The advantages of this approach over preprogrammed machines include adaptability of the same machine to a variety of tasks and the relaxation of the structure of the working environment to allow a more random placement of parts and tools in the assemby area. Other applications, such as machine analysis of chest x-rays and chromosomes, character recognition, fingerprint identification, satellite image interpretation, infrared, optical, and radar target acquisition, and autonomous navigation of rovers for unmanned exploration are largely based on the analysis and interpretation of digital images.

An important aspect in the implementation of scene analysis procedures is the inverse relationship that generally exists between data volume and knowledge about a scene. A single television camera serving as an input to a scene analysis system has, when digitized into a 512 x 512, 8-bit, array a raw data throughput rate that exceeds 62×10^6 bits/sec. Often, however, portions of a scene that contain information relevant to a specific application require a much lower data rate. Consider, for instance, the problem of detecting missing components in a circuit board. In order to detect events of interest in this application, only one frame of video would be required per unit to be inspected. If the inspection system were a matching device that compares an unknown input against a stored prototype image, a weighted and thresholded difference between this prototype and the unknown often suffices to detect significant changes. The difference image can usually be represented with fewer bits, but it contains the necessary information for the solution of this particular problem. In other words, the primary interest in this case is simply the degree of similiarity between the prototype and the input. A decision as to the state of the input unit can be carried out from this information.

In more complex analysis problems, extraction of knowledge from a scene involves a hierarchy of steps which grow progressively more difficult as a function of knowledge gained. Thus, the more we wish to know about a scene, the more complex (in terms of presently known techniques) the process for extracting the desired information. In the example just described, it is evident that detecting missing parts in a circuit board is not a particularly difficult problem. However, counting components, checking their labels, and looking for broken pieces is quite a different matter, involving algorithms which are considerably more complex or may not have practical implementation because of storage and processing requirements.

One of the principal characteristics of scene analysis, in fact, of machine intelligence in general, is the use of heuristics. There is no general theory for selecting the measurements, features, recognition, and interpretation techniques needed for the implementation of scene analysis systems. Although some aspects of these problems have elegant theoretical formulations (e.g., optimum decision rules, search strategies, inductive inference) the state-of-the-art in this field is strictly problem-oriented, where methods are selected based on their performance in a given application.

2.2 A Model of the Scene Analysis Process

As illustrated in Fig. 2.1, the process of digital scene analysis may be divided into four principal categories: (1) sensing and preprocessing, (2) segmentation, (3) recognition, and (4) interpretation. It is important to note that these subdivisions are suggested for convenience and, to a certain extent, by the way in which scene analysis systems in use today have been implemented. It is not implied that human vision and reasoning can be so neatly subdivided nor that these processes are carried out independently of each other. For instance, it is logical to assume that recognition and interpretation are highly interrelated functions in a

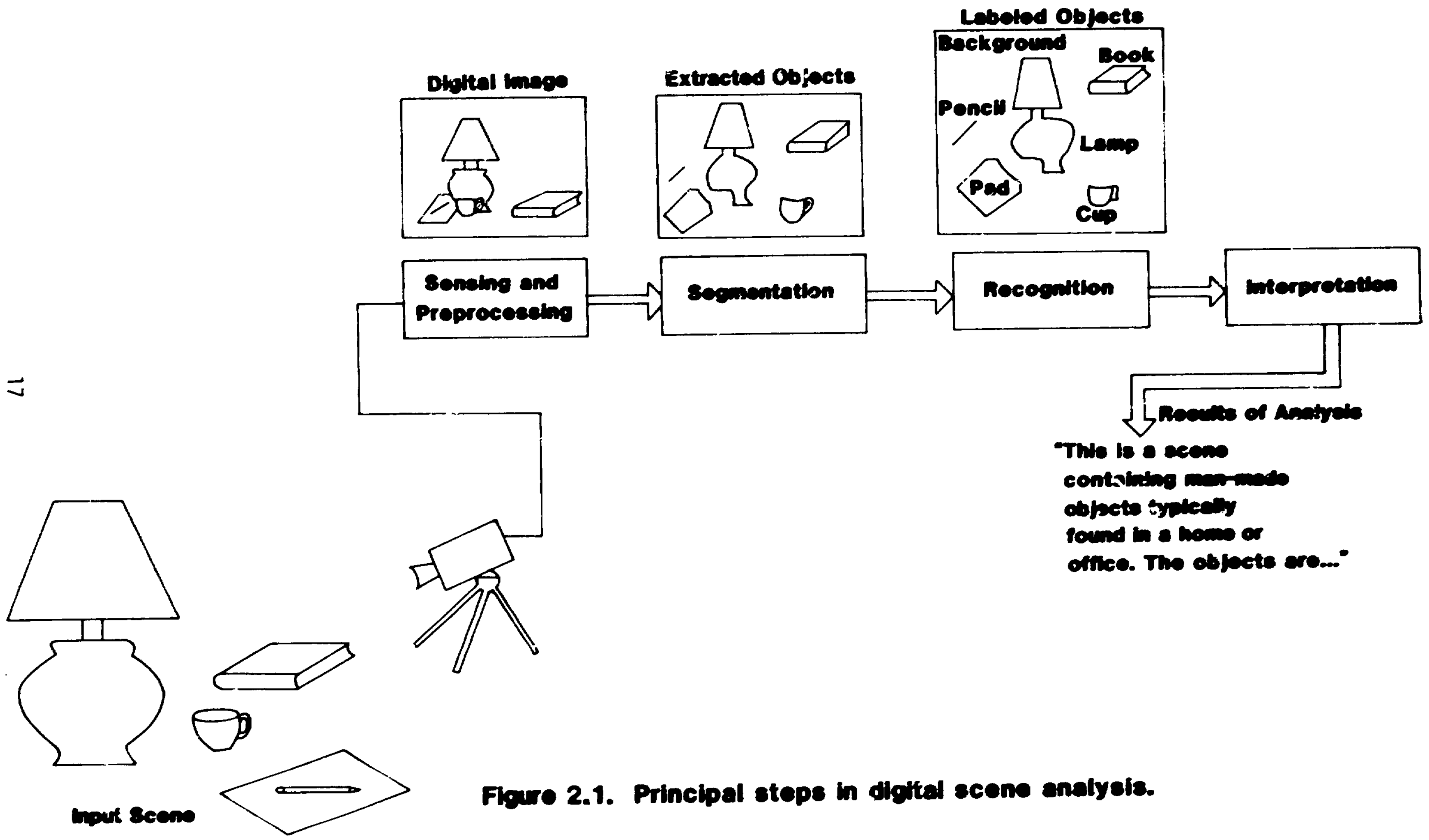


Figure 2.1. Principal steps in digital scene analysis.

human. These relationships, however, are not yet understood to the point where they can be modelled analytically. Thus, the subdivision of functions discussed in this section may be viewed as practical (albeit limited) approach for implementing scene analysis systems given our level of understanding and the analytical tools presently available for the solution of this problem.

The sequence of steps discussed above leads to decomposition of a scene into simpler elements and the arrangement of these elements in the form of a hierarchical representation of the scene. This concept is illustrated in Fig. 2.2 using a simple scene. The highest node in the hierarchy represents the scene itself. At the next level, the scene is composed of several objects: a group of blocks, a pyramid, the wall, and the floor. At the next level, it is shown that the group of blocks is composed of a big block and a little block. The level below this shows that the objects are composed of regions; and finally in the last level, we see that the regions are composed of edge segments, which are numbered in the figure for easy reference.

As implied in the foregoing discussion, obtaining a decomposition of this type can be fairly simple for a human, but is generally quite complicated for a computer. In practice, most procedures for generating a hierarchical representation start at the bottom with the most primitive elements (e.g., the edges) and attempt to arrive at the top node in a bottom-up manner. The basic approach for implementing this bottom-up procedure may be explained with the aid of Fig. 2.3. Starting with a digital representation of a scene derived from one or more sensors, the first problem is to identify and label the primitives in the scene.

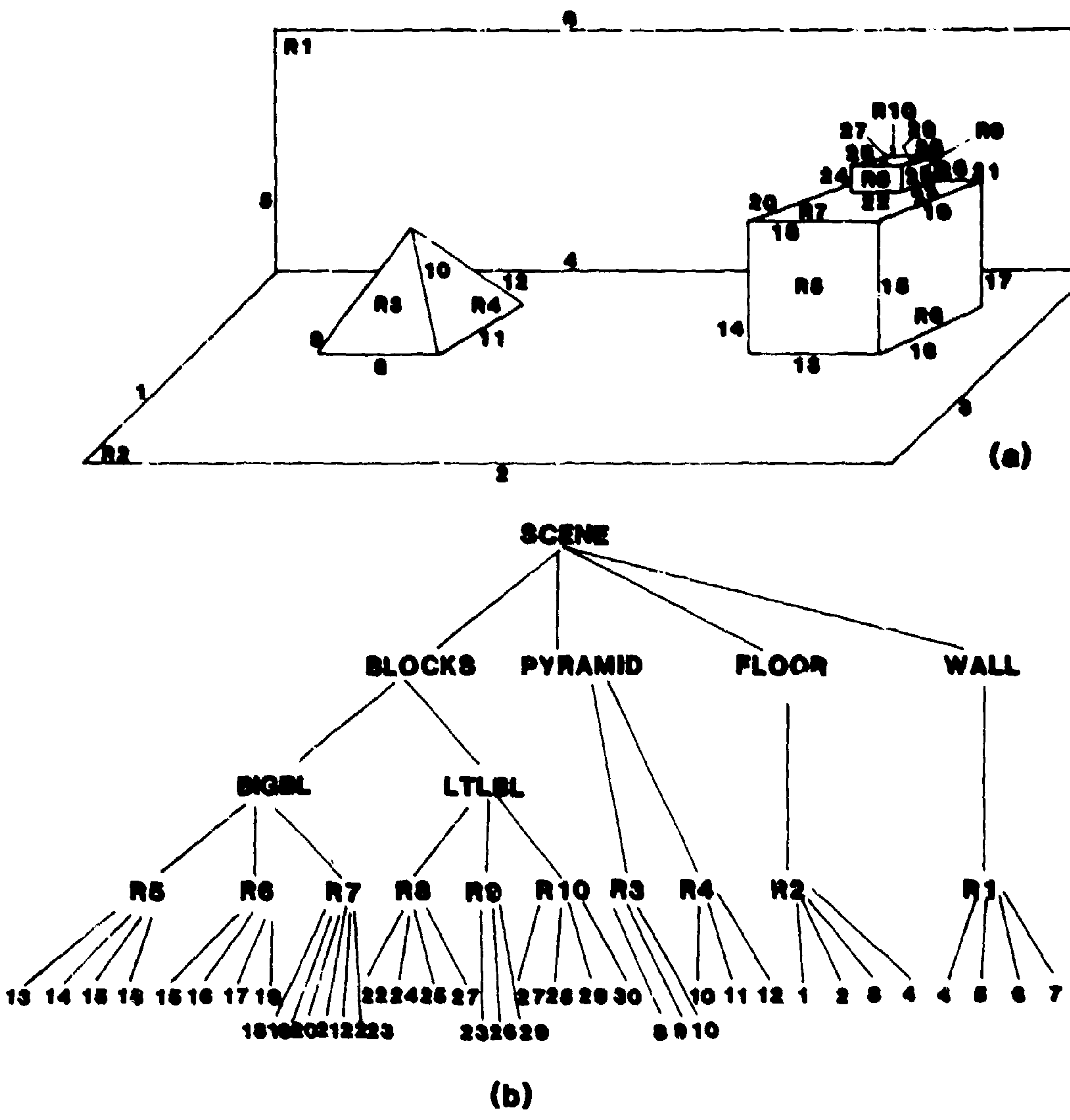


Figure 2.2. A simple scene and its hierarchical representation.

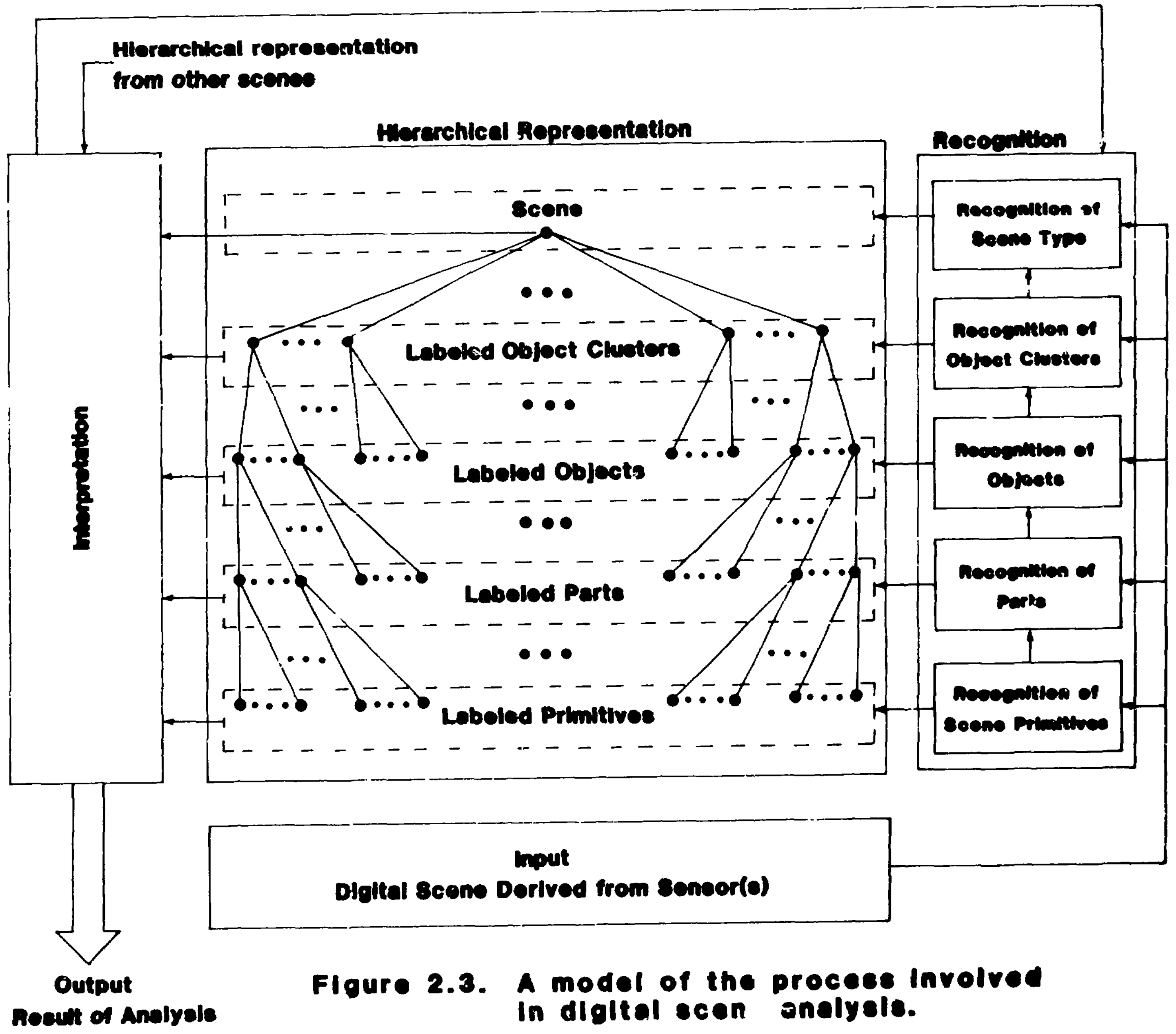

Hierarchical representation from other scenes
Hierarchical Representation
Recognition
Interpretation
Scene
Labeled Object Clusters
Labeled Objects
Labeled Parts
Labeled Primitives
Recognition of Scene Type
Recognition of Object Clusters
Recognition of Objects
Recognition of Parts
Recognition of Scene Primitives
Input
Digital Scene Derived from Sensor(s)
Output
Result of Analysis

Figure 2.3. A model of the process involved in digital scen analysis.

Examples of commonly used primitives are edges, boundaries, and vertices. It is noted that, in Fig. 2.3, the process of assigning a label to a primitive is basically a recognition problem. For instance, if edges are being used as primitives, the first step is to recognize the presence of edges in the scene and to categorize these edges according to type based on descriptors such as length and orientation.

Higher levels in the hierarchy are explained in a similar manner. Labeled parts are obtained by recognition based on the labeled primitives. In turn, these parts are used as inputs to recognition procedures for detecting objects. The resulting labeled objects are classified to form clusters based on some predefined measures of similarity (e.g., man-made objects). The labeled clusters of objects are then recognized as forming a scene of a certain type (e.g., either of military or of non-military interest). Finally, the interpretation stage uses the resulting hierarchical representation, and possibly representations from other scenes, to produce an analysis of the scene.

Figure 2.3 is consistent with the above discussion concerning the four basic processes involved in scene analysis: sensing and preprocessing, segmentation, recognition, and interpretation. In the present model, segmentation techniques are used to extract primitives, parts, and objects from a scene, and recognition algorithms are used to identify (label) these elements. Recognition is also used to group these objects into clusters and to identify the type of scene under consideration. Finally, the int [illegible]etation stage yields an analysis of scene composition and meaning.

The remainder of this chapter is devoted to a detailed discussion of these functions and other aspects related to scene analysis, such as the organization of scene data in a form suitable for processing in the context of the hierarchical model of Fig. 2.3.

2.3 Sensing and Preprocessing

2.3.1 Types of Sensors

The sensing problem is one of converting a physical scene into a form suitable for computer processing. Although a detailed discussion of sensor technology is outside the scope of this section, it is important to briefly discuss the principal types of sensors used in scene analysis applications to give a basic idea of the factors affecting the choice of a particular sensing approach over another.

Sensors are categorized by their response in the electromagnetic energy spectrum (e.g., x-ray, optical, infrared, microwave, and ultrasonic). Often, the choice of an imaging sensor is strictly determined by the environment in which it is expected to operate. For example, underwater applications typically preclude the use of sensors other than low frequency devices. In situations where more than one type of sensor can be used, additional constraints such as resolution, size, and weight play a deciding factor.

Although scene analysis is basically a three-dimensional problem, much of the present work in this area is carried out using planar (image) views. This is due both to limitations in three-dimensional sensor technology and to a lack of procedures for segmentation, recognition, and interpretation of three-dimensional data. Spatial relationships of objects in a scene are approximated by using approaches such as stereo image processing and range imaging [1, 2].

2.3.2 Noise Reduction

One of the principal aspects of preprocessing in practical applications is noise reduction. The purpose of noise reduction is to "clean-up" an image in order to increase the probability of correct segmentation and recognition.

Noise reduction algorithms can be divided into two principal categories: frequency-domain and spatial-domain methods. Frequency-domain techniques are based on the Fourier transform. The procedure is to obtain the Fourier transform of an image, apply a filter (e.g., a bandpass filter) designed to supress frequency regions where the noise is dominant, and then to follow this operation by the inverse Fourier transform to yield an image with a lower noise content. We will not discuss frequency-domain techniques in any detail here because (1) they are well documented in a number of references [3] and (2) they are typically too slow and difficult to implement in practical military systems.

Spatial-domain techniques based on processing small regions in an image are receiving increased attention because they are much simpler to implement in hardware and can often be made adaptive to changing noise characteristics, an important feature in military applications where a system is expected to operate in a variety of unknown conditions. This approach to noise reduction is illustrated below by a recent algorithm due to Lee [4, 5] which is capable of processing images with both additive and multiplicative noise. In addition to its adaptability properties, this algorithm can also be implemented in real time with modest hardware.

2.3.2.1 Additive Noise Filtering

Let $f(x,y)$ be the brightness of a digital image pixel at coordinates (x,y), and let $n(x,y)$ be the value of an additive noise component at those coordinates. The corresponding degrated (noisy) pixel at (x,y) is given by

$$g(x,y) = f(x,y) + n(x,y) \tag{2.3-1}$$

Given $g(x,y)$ and some knowledge about $n(x,y)$ the objective of noise filtering is to obtain an estimate of the true image pixels, denoted by $\hat{f}(x,y)$, which optimizes some criterion. Most current approaches to this problem employ frequency domain techniques, direct invertion, or recursive Kalman filtering [3, 7, 8]. Unlike the algorithm discussed in this section, these approaches are not computationally efficient in terms of real-time processing.

Assume that

$$E[n(x,y)] = \bar{n}(x,y) = 0 \tag{2.3-2}$$

and

$$E[n(x,y)\, n(i,j)] = \sigma^2\, \delta_{xy}\, \delta_{ij} \tag{2.3-3}$$

where δ is the Kronecker delta function (i.e., $\delta_{ij} = 1$ if $i = j$ and 0 if $i \neq j$), E is the expectation operator, and σ^2 is the noise variance. Equation (2.3-2) states that the noise must have zero mean and Eq. (2.3-3) says that the noise must be spatially uncorrelated. The first condition can be easily met by normalization; the second condition is usually true in practice.

Using Eqs. (2.3-2) and (2.3-3), it follows from Eq. (2.3-1) that

$$\bar{f}(x,y) = E[f(x,y)] = E[g(x,y)] = \bar{g}(x,y) \tag{2.3-4}$$

and

$$\begin{aligned} Q(x,y) &\equiv E\{[f(x,y) - \bar{f}(x,y)]^2\} \\ &= E\{[g(x,y) - \bar{g}(x,y)]^2\} - \sigma^2 \\ &= v(x,y) - \sigma^2 \end{aligned} \tag{2.3-5}$$

It is noted that the term $E\{[f(x,y) - \bar{f}(x,y)]^2\}$ is, by definition, the variance of the original, uncorrupted image at coordinates (x,y). Similarly, $v(x,y) = E\{[g(x,y) - \bar{g}(x,y)]^2\}$ is the variance of the noisy image at these coordinates.

A weighted-least-squares estimate of f(x,y) is obtained by considering the criterion function [9]:

$$\begin{aligned} J = 1/2\{[f(x,y) - \bar{f}(x,y)]^2/Q(x,y) + [g(x,y) - \\ f(x,y)]^2/\sigma^2\} \end{aligned} \tag{2.3-6}$$

To determine the estimate, $\hat{f}(x,y)$, that minimizes J we consider the differential

$$\begin{aligned} dJ = df(x,y)\ \{[f(x,y) - \bar{f}(x,y)]/Q(x,y) - [g(x,y) - \\ f(x,y)]/\sigma^2\} \end{aligned} \tag{2.3-7}$$

In order that dJ = 0 for arbitrary df(x,y), the coefficient of df(x,y) in Eq. (2.3-7) must vanish; that is,

$$[\hat{f}(x,y) - \bar{f}(x,y)]/Q(x,y) = [g(x,y) - \hat{f}(x,y]/\sigma^2 \qquad (2.3\text{-}8)$$

where $\hat{f}(x,y)$ is the desired estimate to make the coefficient vanish. Simplification of Eq. (2.3-8) yields

$$\begin{aligned}\hat{f}(x,y) &= \bar{f}(x,y) + \frac{Q(x,y)}{Q(x,y) + \sigma^2}[g(x,y) - \bar{f}(x,y)] \\ &= \bar{f}(x,y) + k(x,y)[g(x,y) - \bar{f}(x,y)]\end{aligned} \qquad (2.3\text{-}9)$$

for all coordinates of M x N image; that is, x = 0, 1, 2, ..., M - 1 and y = 0, 1, 2, ..., N - 1.

In order to interpret this result, it is noted that both $Q(x,y)$ and σ^2 are positive and, therefore, the term $k(x,y) = Q(x,y)/Q(x,y) + \sigma^2$ is bounded between 0 and 1. Since $Q(x,y)$ is the variance of the uncorrupted image pixels, it follows that, for a low signal-to-noise ratio, the noise will dominate so that $k(x,y) \approx 0$ and the estimated $\hat{f}(x,y)$ is approximately equal to $\bar{f}(x,y)$. Conversely, for a high signal-to-noise ratio, $Q(x,y)$ will be much larger than σ^2, $k(x,y) \approx 1$, and the estimate is $\hat{f}(x,y) \approx g(x,y)$, the corrupted pixel at location (x,y).

Implementation of Eq. (2.3.9) is straightforward. Since, from Eq. (2.3-4), the means of the original and noisy images are assumed equal, it follows that $\bar{f}(x,y)$ can be obtained by computing the average value of the noisy image, $\bar{g}(x,y)$, at coordinates (x,y). Similarly, $Q(x,y)$ is obtained by computing the difference between the variance of the noisy image at (x,y) and the noise variance, σ^2. In other words, letting $\bar{f}(x,y) = \bar{g}(x,y)$ it follows from Eqs. (2.3-5) and (2.3-9) that

$$\hat{f}(x,y) = \bar{g}(x,y) + \frac{v(x,y) - \sigma^2}{v(x,y)}[g(x,y) - \bar{g}(x,y)] \qquad (2.3\text{-}10)$$

for x = 0, 1, 2, ..., M - 1 and y = 0, 1, 2, ..., N - 1.

The approach followed in obtaining $\bar{g}(x,y)$ and $v(x,y)$ is to use the local mean and variance computed in a (2n + 1) x (2m + 1) window centered at (x,y). That is,

$$\bar{g}(x,y) \approx \frac{1}{2(n+1)(2m+1)} \sum_{k=x-n}^{x+n} \sum_{q=y-m}^{y+m} g(k,q) \qquad (2.3\text{-}11)$$

and

$$v(x,y) = \frac{1}{(2n+1)(2m+1)} \sum_{k=x-n}^{x+n} \sum_{q=y-m}^{y+m} [g(k,g) - \bar{g}(k,g)]^2 \qquad (2.3\text{-}12)$$

The procedure is to compute $\bar{g}(x,y)$ and $v(x,y)$ from the known image $g(x,y)$ for x = 0, 1, 2, ..., M - 1 and y = 0, 1, 2, ..., N - 1. The desired es mate $\hat{f}(x,y)$ then follows directly from Eq. (2.3-10) by using a given value of σ^2.

T e two principal factors affecting the quality of the estimate $\hat{f}(x,$ ' are window size and the value of the noise variance. If the window is too small, noise filtering will not be effective because, from Eq. (2.3-11), $\bar{g}(x,y)$ will be approximately equal to $g(x,y)$ and $\hat{f}(x,y) \approx g(x,y)$ in Eq. (2.3-10). If the window is too large, the estimate will be a smoothed (blurred) version of the original. Lee [4] reported that a 7 x 7 window appears to be a good compromise. Our own results verify his conclusion concerning window size. The effects of σ^2 are discussed below and in the following section.

In order to investigate the suitability of Eq. (2.3-10) for noise reduction, we conducted a number of experiments using FLIR images [10] corrupted by computer-generated noise as well as practical FLIR images obtained in the field under realistic conditions. Figure 2.4(a) shows an infrared image containing a simple target and Fig. 2.4(b) shows the same image severely corrupted by Gaussian noise with zero mean and $\sigma^2 = 300$. The result obtained with Eq. (2.3-10) using a 7 x 7 window and the above known value for the noise variance is shown in Fig. 2.4(c). The improvements over the noisy image are evident in this picture.

As a more realistic example, Fig. 2.5(a) shows a typical FLIR image containing two targets and Fig. 2.5(b) is the result obtained using Eq. (2.3-10) with a 7 x 7 neighborhood and an empirically determined value of $\sigma^2 = 60$. Although the image was improved considerably, it is of interest to note that the area above the leftmost target is still quite visibly corrupted. The reason for this is that σ^2 is truly unknown for this image and our estimate of $\sigma^2 = 60$ was not good enough for this particular area of the image; that is, the noise variance is not only unknown but it is also spatially variant. An adaptive technique for estimating the noise variance throughout an image is discussed in the next section.

2.3.2.2 Adaptive Estimation

An effective implementation of Eq. (2.3-10) for automatic image processing applications requires that σ^2 be known with a reasonable degree of accuracy. Since this parameter is often difficult to estimate in practice for the full range of operating conditions of most sensors, it

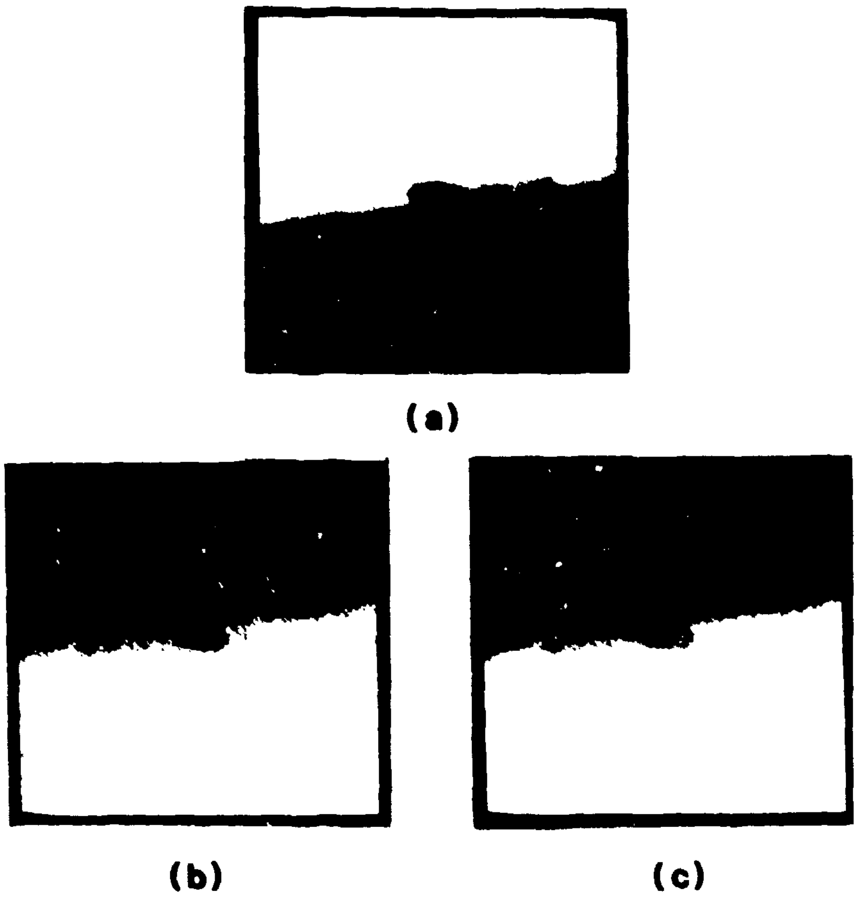
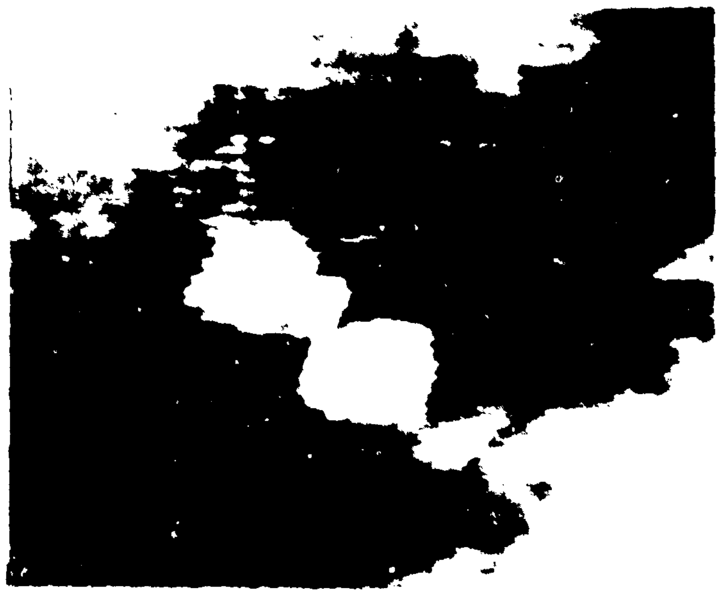


Figure 2.4. (a) Original, (b) image corrupted by additive Gaussian noise, (c) result obtained using Eq. (2.3-10) (Original image courtesy of Dr. Lewis G. Minor, U. S. Army, Huntsville, Alabama.).

(a)



(b)

Figure 2.5. (a) Original, (b) result obtained using Eq. (2.3-10) with a 7 x 7 window and $\sigma^2 = 60$. (Original image courtesy of the Equipment Group, Texas Instruments, Inc., Dallas, Texas).

is of interest to have available a procedure which will estimate the noise variance for a particular input image, thus yielding an adaptive implementation of Eq. (2.3-10).

The principle behind the procedure discussed below is as follows: If $f(x,y)$ in Eq. (2.3-1) were constant, the variance obtained from $g(x,y)$ would in fact be the variance of the noise. When $f(x,y)$ is not constant, the variance obtained from $g(x,y)$ is greater, being influenced by the variance of both $f(x,y)$ and $n(x,y)$. When variance estimates are computed about a neighborhood of a pixel located at (x,y), the variance of the noise may be approximated by the smallest of the variances obtained at each pixel in the neighborhood of (x,y). In other words, the smallest variance should occur when $f(x,y)$ is constant and is, therefore, an estimate of the noise variance at (x,y).

This idea was employed in developing an adaptive estimator of the noise variance. In each 7 x 7 window located at (x,y), $x = 0, 1, 2, \ldots, M - 1$, $y = 0, 1, 2, \ldots, N - 1$, the variances of all neighbors of (x,y) in the window were computed in a 7 x 7 window about the point in question. That is, for any point (s,t) inside the window centered at (x,y) we defined a 7 x 7 window centered at (s,t) and computed the variance of the points inside this new window. This procedure yielded 49 variance estimates around every point (x,y) in an image. In order to allow for variations due to the small size of the window, the average of the 5 smallest variances was computed and designated as the variance σ^2 for use in Eq. (2.3-10) at location (x,y) in the image.

The results obtained with this technique were considerably better than those obtained using a constant value for σ^2, as illustrated by the images shown in Fig. 2.6. Part (a) of this figure shows the same original considered in the previous section and Fig. 2.6(b) is the result obtained with the adaptive variance estimator. By comparing this image with Fig. 2.5(b) it is noted that noise reduction in the area above the left-most target was effectively accomplished using the adaptive method. As another illustration of the power of this technique, Fig. 2.6(c) shows a more complex FLIR image and Fig. 2.6(d) is the result after processing with the adaptive variance estimator. The improvement of the processed images over the originals is quite clear in these results.

2.3.3 Enhancement

Another important preprocessing function in scene analysis is enhancement. As in the case of noise reduction, enhancement techniques may also be divided into frequency-domain and spatial-domain methods [3]. The approach in the frequency domain is exactly the same as the one discussed in Section 2.3.2, with the exception that filters are chosen to highlight a given frequency range (e.g., high-frequency emphasis for image sharpenning). The limitations of frequency domain discussed in Section 2.3.2 also apply to enhancement algorithms based on this approach.

Many spatial domain techniques for image enhancement have the potential for real-time implementation. The methods discussed in this section are based on histogram processing techniques. These methods are illustrative of algorithms which can be implemented in real time and which have the important property of automatic adaptability to changing scene conditions.

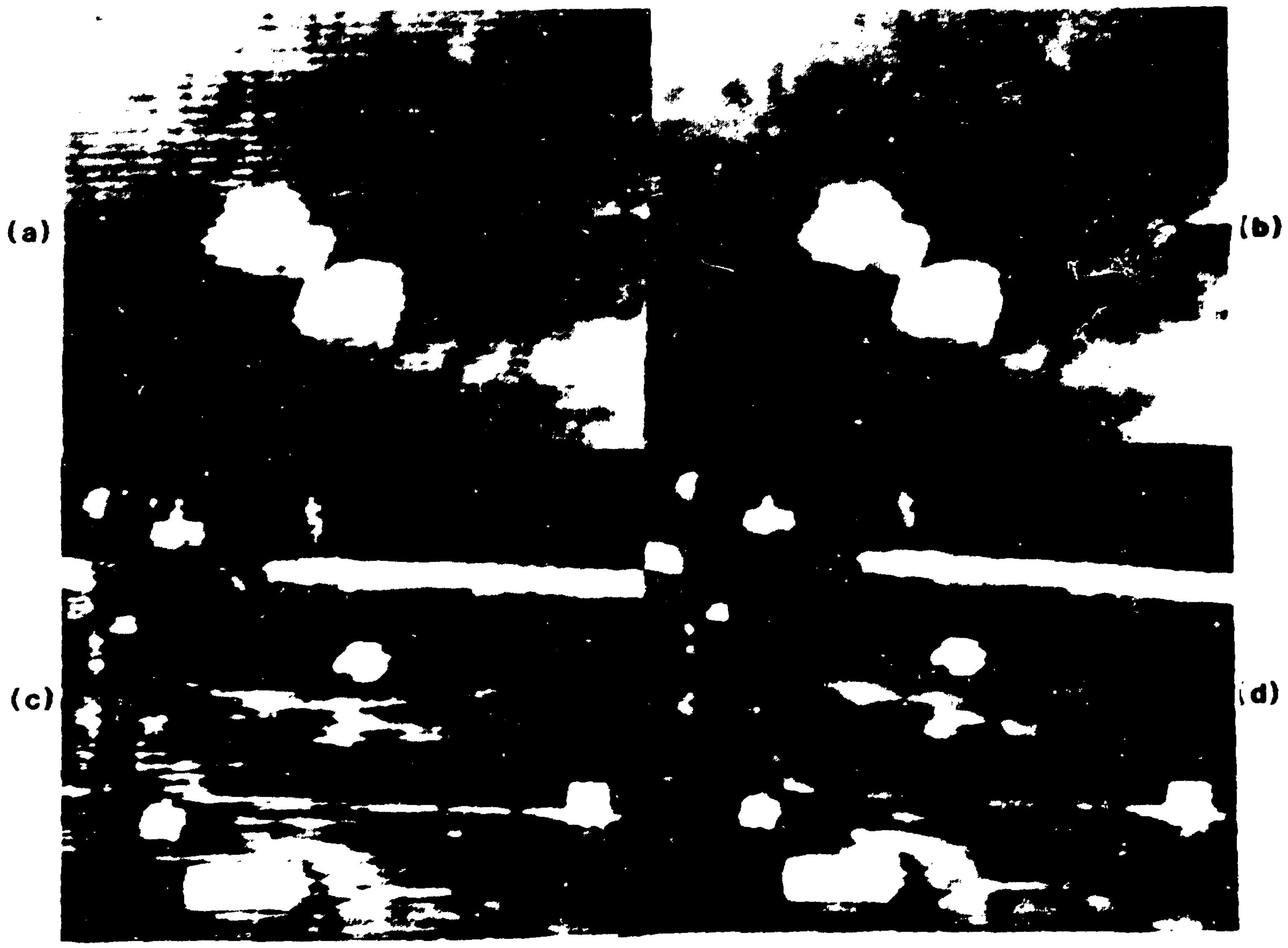


Figure 2.6.(a) and (c) are original FLIR images and (b) and (d) are the corresponding results obtained by using Eq. (2.3-10) with adaptive variance estimation. (Original images courtesy of the Equipment Group, Texas Instruments, Inc., Dallas, Texas).

A histogram of the gray levels in an image provides a global description of its appearance. The two principal methods of histogram processing are histogram equalization (also called histogram linearization or histogram flattening) and histogram specification [11-17].

In histogram equalization, the approach is to transform the pixels of the input image so that they will have a flat (uniform) histogram. The motivation for this type of processing is to increase the dynamic range of the intensity values of an image and thus improve its overall appearance.

Histogram specification techniques attempt to transform the pixels in an image so that their histogram will have a pre-specified form. There are two basic approaches used in histogram specification. The first is to use interactive techniques where a human operator provides a historam shape and evaluates the results by trial and error. The second involves the use of a priori knowledge about what the resulting histogram should be. An example of this is found in computer processing of chest radiographs. Chest x-rays that have been developed properly have characteristic histograms which may be obtained by averaging the histograms of a set of samples. This average histogram can then be used as the norm for mapping, by means of histogram specification techniques, the intensity values of a given x-ray which has been improperly developed.

2.3.3.1 Histogram Equalization

Let r represent the intensity of the input pixels. Assume for a moment that r is a continuous variable in the range $0 \leq r \leq 1$ and with probability density function $p_r(r)$. If r is transformed to a new intensity variable s by means of the transformation function

$$s = T(r) \tag{2.3-13}$$

it follows from probability theory that

$$p_s(s) = \left[p_r(r) \frac{dr}{ds} \right]_{r = T^{-1}(s)|} \tag{2.3-14}$$

where it is assumed that T(r) is a monotonic function.

In histogram equalization, we choose

$$s = T(r) = \int_0^r p_r\,(w)dw \tag{2.3-15}$$

where w is a dummy variable of integration. By substituting (2.3-15) into (2.3-14) we obtain

$$p_s(s) = \begin{cases} 1 & 0 \leq s \leq 1 \\ 0 & \text{elsewhere} \end{cases} \tag{2.3-16}$$

which is seen to be a flat or uniform density.

In the discrete case, $p_r(r)$ is approximated by

$$p_r(r_k) = \frac{n_k}{n} \tag{2.3-17}$$

where n is the total number of pixels in the image and n_k is the number of pixels with discrete intensity value k. The transformation for histogram equalization then becomes

$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{n} \quad k = 0, 1, \ldots, L - 1 \qquad (2.3\text{-}18)$$

where L is the number of discrete levels in the image. In practice, $p_s(s_k)$ is an approximation to $p_s(s)$ and, consequently, it is seldom perfectly uniform.

Based on the above discussion, we see that histogram equalization is straightforward. It consists of mapping each input pixel with value r_k into a pixel with value s_k, where s_k is given by Eq. (2.3-18).

2.3.3.2 Histogram Specification

The procedure for histogram specification is slightly more complicated. To see how this can be accomplished, let us return for a moment to continuous gray levels, and let $p_r(r)$ and $p_z(z)$ be the original and desired probability density functions, respectively. Suppose that a given image is first histogram equalized using Eq. (2.3-15); that is

$$s = T(r) = \int_0^r p_r(w)\, dw \qquad (2.3\text{-}19)$$

If the desired image were available, its levels could also be equalized by using the transformation function

$$v = G(z) = \int_0^z p_z(w)\, dw \qquad (2.3\text{-}20)$$

The inverse process, $z = G^{-1}(v)$, would then yield the desired levels back. This, of course, is a hypothethical formulation since the z levels are precisely what we are trying to obtain. It is noted, however, that

$p_s(s)$ and $p_v(v)$ would be identical uniform densities since the final result of Eqs. (2.3-14) through (2.3-16) is independent of the density. Thus, if instead of using v in the inverse process we use the uniform levels s obtained from the original image, the resulting levels, $z = G^{-1}(s)$, would have the desired probability density function. Assuming that $G^{-1}(s)$ is single-valued, the procedure can be summarized as follows:

(1) Equalize the levels of the original image using Eq. (2.3-15).

(2) Specify the desired density function and obtain the transformation function $G(z)$ using Eq. (2.3-20).

(3) Apply the inverse transformation function, $z = G^{-1}(s)$ to the levels obtained in Step 1.

This procedure yields a processed version of the original image where the new gray levels are characterized by the specified density $p_z(z)$. A discrete formulation of this method parallels the development of Eqs. (2.3-17) and (2.3-18), and the mapping $G^{-1}(s)$ is accomplished in the discrete case by rounding $G^{-1}(s)$ to the nearest allowable discrete pixel value.

Figure 2.7 illustrates the enhancement capabilities of the histogram equalization technique, which is automatic since no operator interaction is required. The histogram of the images on the left side of Fig. 2.7 would be expected to span two relatively narrow gray-level ranges; the pixels of the corresponding enhanced images on the right side of Fig. 2.7 span a considerably larger spectrum of the gray-level scale. As is characteristic of this method, a simple spreading of the gray-level histogram can have a remarkable effect on the output image.

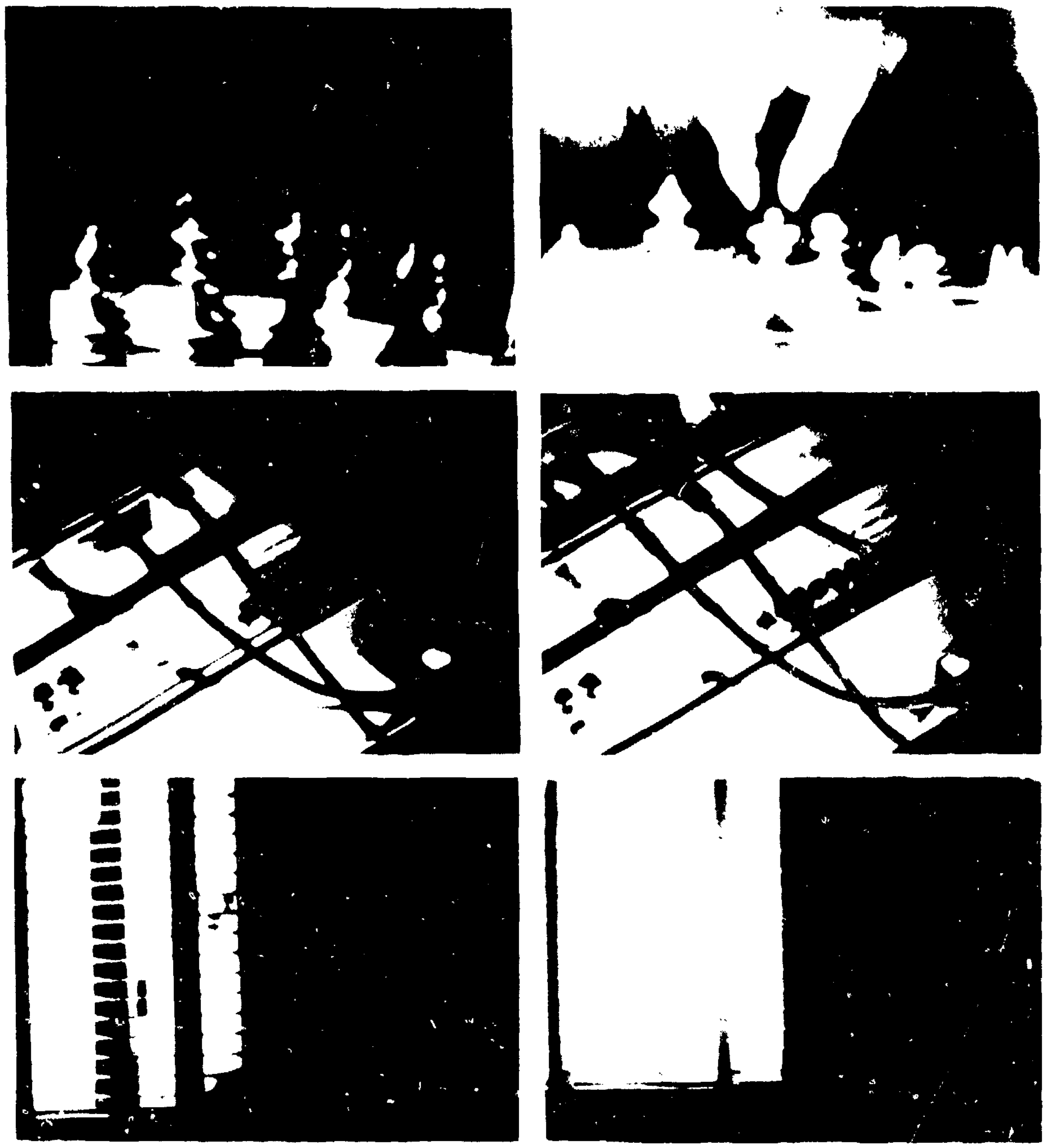


Figure 2.7. Examples of images before (left) and after (right) histogram equalization.

Although, as shown in Fig. 2.7, histogram equalization can yield excellent enhancement results, this approach often fails to bring out high contrast details in an image. An example of this is given in Fig. 2.8. The image on the top left is the original and the image on the top right is the result after histogram equalization. For all practical purposes, this result is of little use because it failed to bring out the obscure side of the knight. The image displayed at the bottom of Fig. 2.8 was obtained by interactive histogram specification [17]. It is noted that the detail on the obscure side of the knight was vividly brought out by this approach.

2.4 Segmentation

Segmentation is the process that breaks up a sensed scene into its constituent parts or objects. Virtually hundreds of segmentation algorithms have been proposed in the literature over the past fifteen years [18-20]. Not surprisingly, this is still an active area of research because of its importance in any practical scene analysis application. Since what we as humans define as objects in a scene is the result of mental processes that are not well understood, present analytical techniques for extracting structural information from pictorial data are necessarily heuristic in nature and very much application oriented. The segmentation problem is further compounded by a lack of parallel processing approaches which would allow the system to perceive a scene in a global way. Thus, most segmentation algorithms in use today are oriented toward processing small regions in an image, with the limiting special case of point-by-point processing techniques.

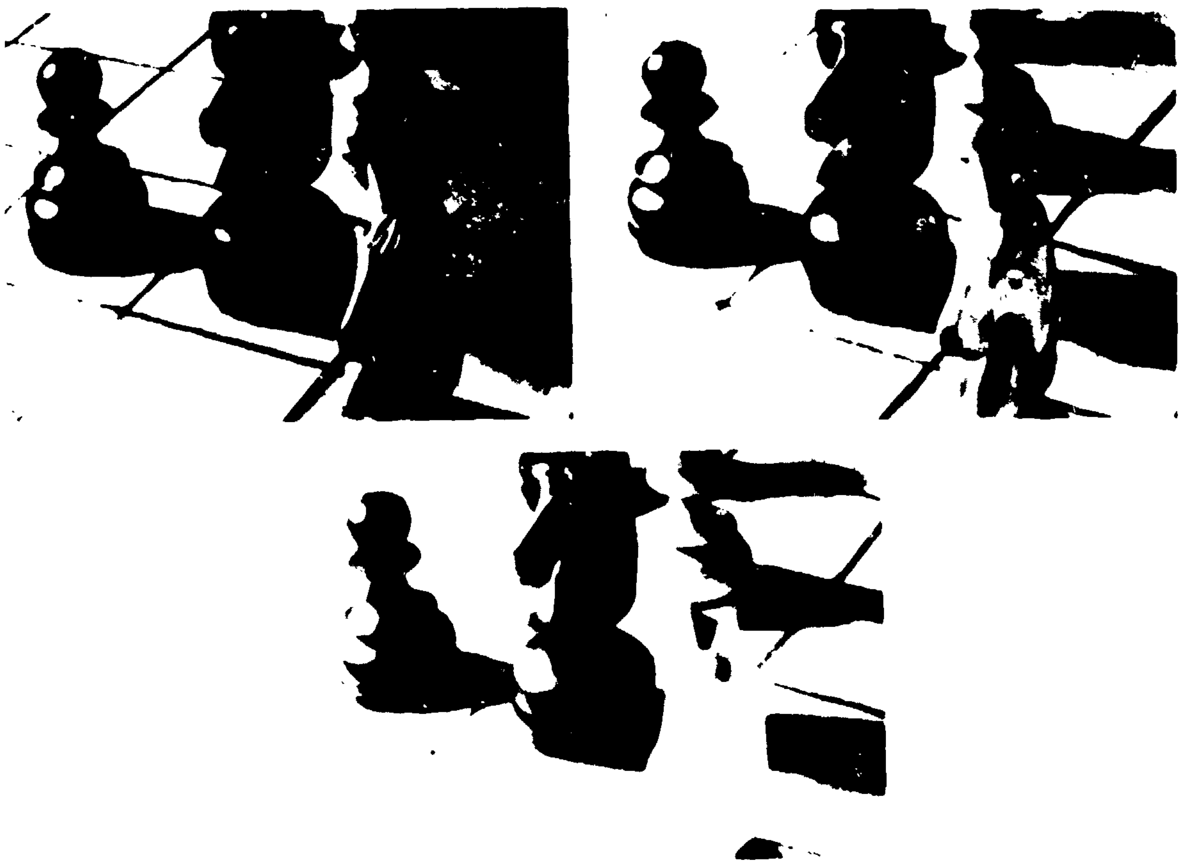


Figure 2.8. Original image (top left) and results after histogram equalization and histogram specification.

Algorithms for segmentation range from thresholding [21, 22] and line detection techniques [3], to region growing, merging, and splitting [23, 24] using features such as texture and local contrast differences. Attempts to incorporate some degree of contextual information into the segmentation process include the use of relaxation techniques [25], plan-guided analysis [26 - 28], and the use of semantic information [29].

Basically, segmentation techniques may be classified as either edge-based or region-based [3, 30, 31]. The former extracts edges according to the local properties of an image, such as gradient or Laplacian properties and is often sensitive to local noise. The edges produced are usually broken and difficult to use for object location. The region-based methods exploit the global properties of the images in segmentation (e.g., histogram analysis), and are more stable. An advantage is that edges which are region boundaries are naturally closed. The closed edges are often also more suitable for shape description or vertex location. However, pert.. t region segmentation is rarely obtained through global methods such as histogram analysis. The principal problem is that false regions may also be detected along with the true object regions due to the imperfection of a threshold selection or variations in scene illumination.

Previous .ffort. \ segmentation using local edge information have used a threshold selected from the image histogram according to the Laplacian properties of an image [21]. This technique was reported to be successful for bim(histogram segmentation. However, the selection of multiple thresholds on a multimodal histogram is often crucial in scene segmentation. The mode method [32] in which the thresholds are set at the minima between the modes of the histogram has been used successfully in segmenting images of white blood cells. The method involves histogram smoothing, mode searching and threshold setting. For natural scenes, too many regions are often segmented if the histogram has not been smoothed and too few regions are detected if the histogram has been overly smoothed. There are two ways to improve the segmentation results, splitting the regions [22, 33] due to oversmoothed histograms or merging the false regions due to undersmoothed histograms [23]. The former is a top-down segmentation method. The regions resulting from oversmoothing must be divided or split into subregions. The latter is a bottom-up segmentation method; the regions resulting from undersmoothing the histogram must be merged together. Either the top-down or the bottom-up approach will improve the segmentation results of the mode method.

For a scene with solid objects, the edge intensity between two regions located on the same surface in the scene is much weaker than that between two different surfaces. Based on this property, two regions with weak adjacent edge intensity are very likely to be on the same surface and need to be merged. On the other hand, two regions with strong adjacent edge intensity are likely to be on different surfaces and should remain separate. Therefore, the local edge intensity at the adjacent boundaries of two regions can be used as a merge criterion for bottom-up segmentation. The edges which have magnitude greater than a preselected percentile on the edge intensity histogram are considered as strong edges. A group of regions being merged are said to be in an equivalence class under this merge criterion.

The basic techniques involved in segmentation are illustrated in this section by means of a procedure which combines global and local information. The technique, called Global-Local-Edge-Coincidence (GLEC), is basically a split-and-merge segmentation method. The results of GLEC appear equivalent to selecting thresholds on a multimodal histogram; however, the processing method is designed to find the equivalence classes of the atomic regions which are generated by an undersmoothed histogram segmentation method according to a reference map consisting of the "expected" region boundaries. The expected boundaries are located using local edge information. The reference map is produced by binary correlation of the strong local edges with the atomic region boundaries of the image. Although edges are used to discuss the map generation, other features in an image, such as region contrast and texture similarity, may also be used to generate the reference map.

2.4.1 Segmentation Using a Reference Map

Image segmentation can be considered as a problem of partitioning an image into subsets by defining an equivalence relation on the pixels in the image array such that all the pixels in a region will be in an equivalence class [33, 34]. To segment an image from the given region boundary map is equivalent to partitioning the non-boundary pixel subset of an image by using an equivalence relation. Consider a set of boundary pixels B_b and a set of non-boundary pixels B_n in an N X M array S. Then

$$S = B_b \cup B_n \qquad (2.4\text{-}1)$$

If a subset R_1 of B_n such that the elements p_{i1}, p_{i2} ..., $p_{i\ell}$ in R_i are in an equivalence class, i.e., $p_{i1} \sim . . . \sim p_{i\ell}$ where $\sim$ is an equivalence relation among the non-boundary pixels in B_n, then p_{i1}, p_{i2}, ..., $p_{i\ell}$ are the interior pixels of region i. If k equivalence classes can be partitioned in B_n, then there will be k regions in the image S and

$$B_n = \bigcup_{i=1,k} R_i \tag{2.4-2}$$

An equivalence relation can be defined on the non-boundary pixels: two pixels p and q are said to be equivalent to each other if no edges can be found between them. It is obvious that all the interior pixels of a region will be in an equivalence class. After all the interior pixels of the regions are determined, the segmented image can be obtained by splitting and merging the boundary pixels in B_b to all the regions R_i. A boundary pixel is merged to a region which has more interior pixels adjacent to it. If two regions have the same number of adjacent pixels to a boundary pixel, then this pixel will be merged to the one with highest priority. The merging priority is assigned according to a clockwise sequence, as shown in Fig. 2.9.

The foregoing concepts may be illustrated by the following example. Figure 2.10 is a given region contour map. The shaded area in Fig. 2.10 represents the given region contour of the image. If the non-boundary pixels are labelled according to a top-bottom and left-right manner, it is clear that pixel 1 is equivalent to pixel 2 and is also equivalent to pixel 3 since the equivalence relation is transitive. Also note that there are three equivalence classes in Fig. 2.10 which

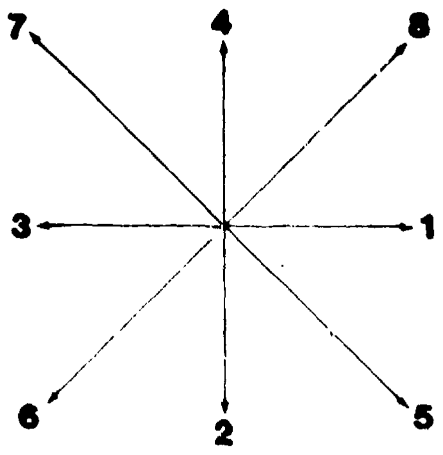


Figure 2.9. The merging priority.

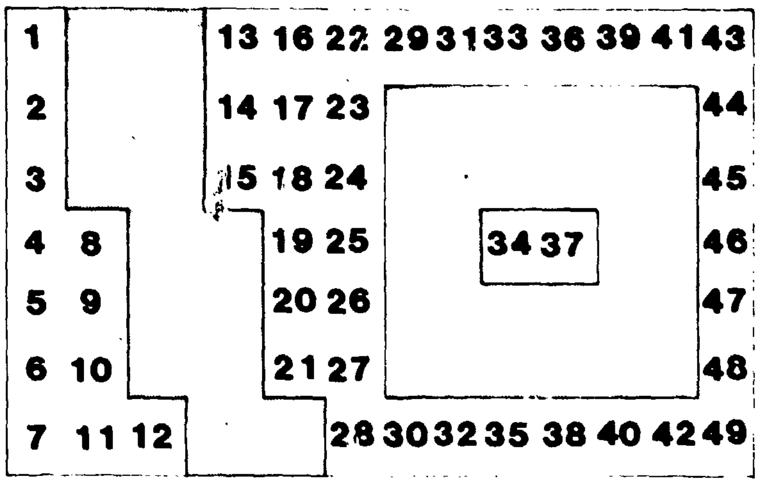


Figure 2.10. Example of segmentation by using a given region contour map. The shaded area are the region contours.

have been relabelled in Fig. 2.11. The final segmented results can be obtained by splitting and merging the given contour of Fig. 2.11, as shown in Fig. 2.12. This result shows that if the region boundaries can be determined from an image, then segmentation may be reduced to a problem of finding the equivalence classes by using these boundaries as a reference. In the next section, a method using Global-and-Local-Edge-Coincidence will be described to generate the expected region boundaries of the objects in the scene.

2.4.2 Generating the Boundary Map Using Global-Local-Edge-Coincidence

In a scene with solid objects, the edge intensity between two regions located on the same surface of an object is much weaker than that between two different surfaces. Based on this property, the edges with high intensity (large gradient values) are the most likely object boundaries. A comparison study of several edge detectors [35] showed that the 10 to 20 percentile of high-intensity local edges are meaningful and visually perceptible in many images. Also, the majority of object boundaries are among the meaningful edges. Selecting a percentile in the edge intensity histogram to obtain all the object boundaries is scene-content dependent. A trial and error method is usually required. Whenever a threshold is set in the edge-processed image, some of the local line edges will also be segmented along with the object boundaries. Also, the edges corresponding to the object boundaries in the thresholded edge map may not be the closed edges due to nonuniform surfaces or uneven illumination of the object in the scene. For broken boundaries, the segmentation method described in Section 2.4.1 will generally fail because the

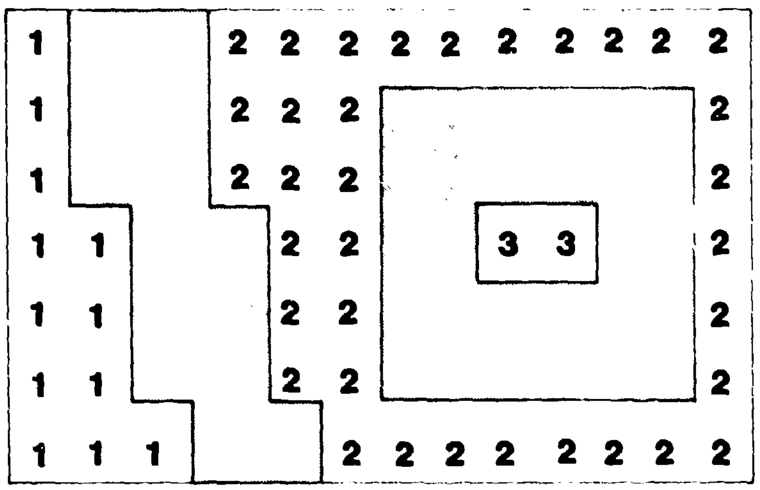
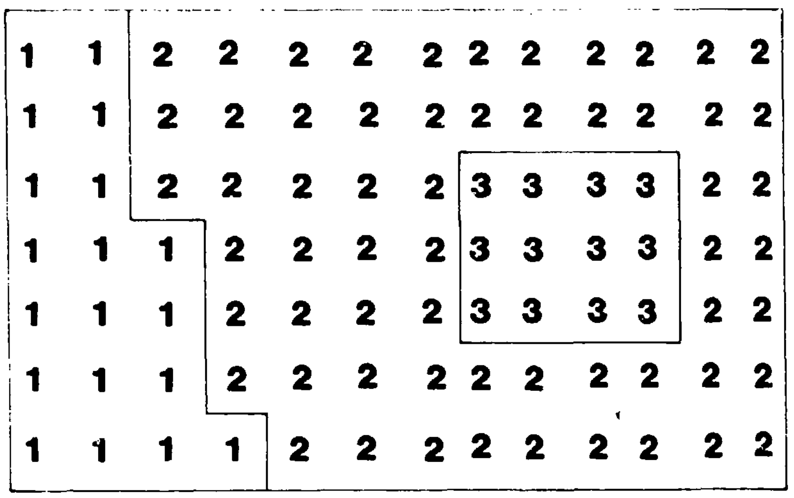


Figure 2.11. The three equivalence classes of Fig. 2.10.



1	1	2	2	2	2	2	2	2	2	2	2	2
1	1	2	2	2	2	2	2	2	2	2	2	2
1	1	2	2	2	2	2	3	3	3	3	2	2
1	1	1	2	2	2	2	3	3	3	3	2	2
1	1	1	2	2	2	2	3	3	3	3	2	2
1	1	1	2	2	2	2	2	2	2	2	2	2
1	1	1	1	2	2	2	2	2	2	2	2	2

Figure 2.12. The final segmented results by splitting and merging the shaded area of Fig. 2.10.

nonboundary pixels of two surfaces may be classified as one equivalence class due to the transitive relation at the broken boundaries.

One of the common features in a scene with solid objects is that the area of the object boundaries is usually smaller than that of the surfaces. The image intensity of the object boundaries often corresponds to the minima in the intensity histogram [32]. Therefore, the mode method is applicable for segmenting the object boundaries; however, the threshold setting of the histogram depends on the degree of smoothing. In addition, even if an appropriate threshold segments a meaningful region in an image, it may also generate a false region at another part of the image. For instance, the window of a building may be segmented by setting a certain threshold in the histogram; a false region may also be generated on the roof due to the same threshold. However, if the histogram is undersmoothed, too many regions will be generated due to over-specifying the thresholds in the histogram; the real object boundaries are still contained in the segmented region boundaries or portions of the region boundaries of certain regions. The object boundaries will be generated if those false region boundaries can be eliminated. In order to avoid ambiguity, the regions which are segmented by using the mode method are called <u>atomic regions</u>. The boundaries of the atomic regions are called <u>atomic region boundaries</u> or <u>global edges</u> in the following discussion.

Since both the global edges and the strong local edges contain the real object boundaries, it is appropriate to use this information to produce the "expected" region boundary map. Experimental results show that most of the object boundaries corresponding to global edges will match the strong local edges in the vicinity of one pixel. By using this property, the region boundary map may be determined in the following way.

All the atomic regions are first labelled according to a top-bottom and left-right sequence. Boundaries of all the atomic regions are then detected and correlated with the local edge information. Assume that $R(i,j)$ is a region-image array and $E(i,j)$ is a local-edge array. Let $R(i,j) = 1$ for all the atomic boundary points in R and $E(i,j) = 1$ for all the edge points in E if $E(i,j) > T$. All other entries in R and E are zero.

Then a binary reference map C_T is generated according to the following calculation:

$$\begin{aligned} C_T(i,j) = R(i,j) \cdot [&E(i,j-1) + E(i+1,j) + E(i-1,j) \\ &+ E(i,j+1)] \end{aligned} \tag{2.4-3}$$

where $\cdot$ is the logical AND and + is the logical OR. It is noted that $C_T(i,j)$ is the binary cross correlation result of the atomic region boundaries and the intensity thresholded local edges with their four adjacent neighbor pixels. T is a selected threshold of edge intensity in the edge-processed image.

Next, a coincidence test of the adjacent edges between the atomic regions is used to produce the expected region boundary map and provide the merge information for segmentation. Consider A_{uv} as the adjacent edge between region u and region v consisting of ℓ boundary points, such that

$$A_{uv} = \{p_1, p_2, \ldots, p_\ell\} \tag{2.4-4}$$

Let

$$Q = \frac{1}{\ell} \sum_{i-1}^{\ell} C_T\, p_i \qquad (2.4\text{-}5)$$

Then Q is the percentage of match of the adjacent edge A_{uv} with the reference C_T. The entire adjacent edge of the atomic regions will be retained in the expected region boundary map if $Q > T_C$. Atomic region u and atomic region v are assumed to be in the same equivalence class and will be merged together if $Q \leq T_C$, where T_C is a number that defines the acceptable threshold of the percentage of match in the coincidence test. This test is conducted on all the adjacent edges of the atomic regions until the expected region boundary map is produced. Then the segmented picture may be produced by finding the equivalence classes using this generated region boundary map as a reference.

The procedure for generating the region boundary map using the global-local-edge-coincidence may be illustrated by the following example. Assume that four atomic regions are segmented, as shown in Fig. 2.13. The shaded area shows the atomic region boundary pixels. The boundary pixels adjacent to ab, bc, de, df are the adjacent edges of region 1 and region 3, region 1 and region 2, region 2 and region 4, region 3 and region 4, respectively. Further, assume that the strong edges of the image are shown in Fig. 2.14. Then Fig. 2.15 shows the result of binary correlation of Fig. 2.14 with the atomic region boundaries of Fig. 2.13. Figure 2.16 shows the expected region boundary map after the GLEC test. Using this boundary map as the reference, region 1 will be merged with

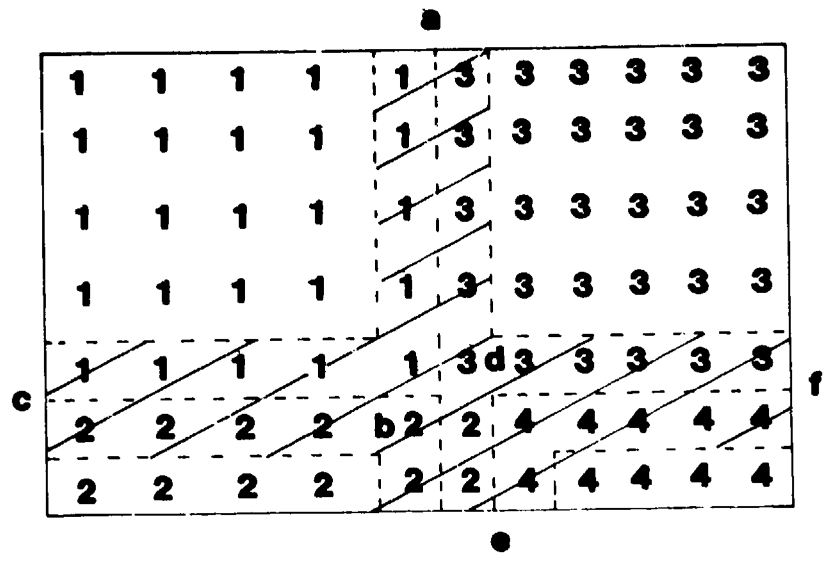


Figure 2.13. Four atomic regions are shown in this figure. ab,bc,de, and df are the adjacent edges between region 1,3; 1,2; 2,4; and 3,4, respectively.

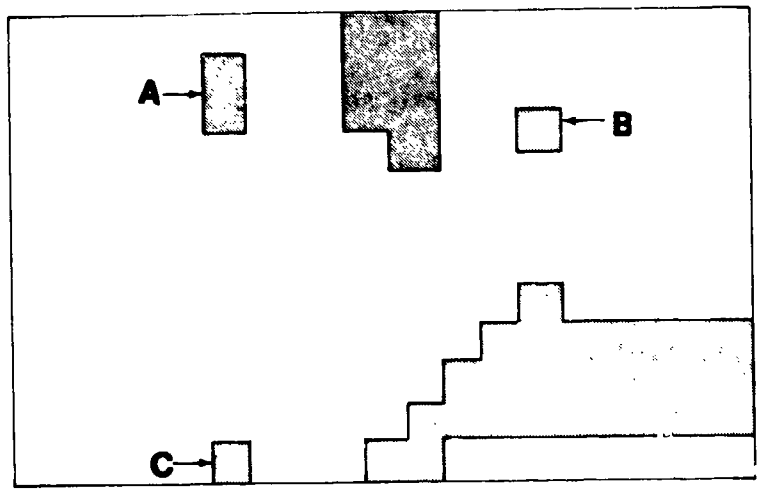


Figure 2.14. The shaded area shows the strong edges.

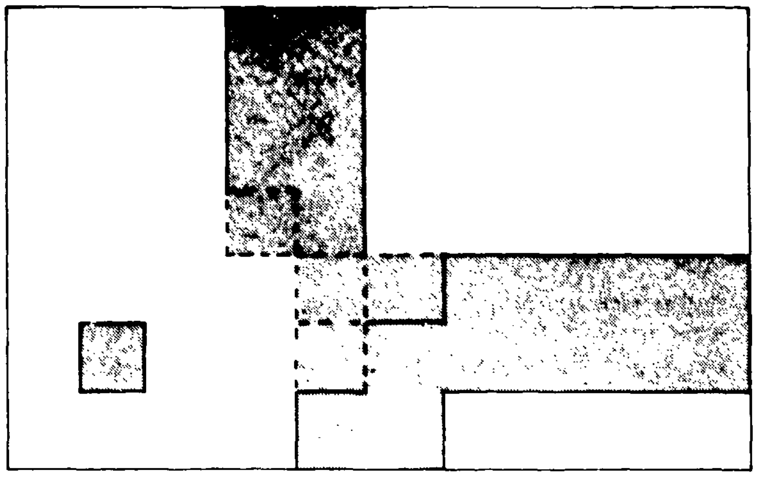


Figure 2.15. The result of binary correlation of the atomic region boundaries with the strong edges.

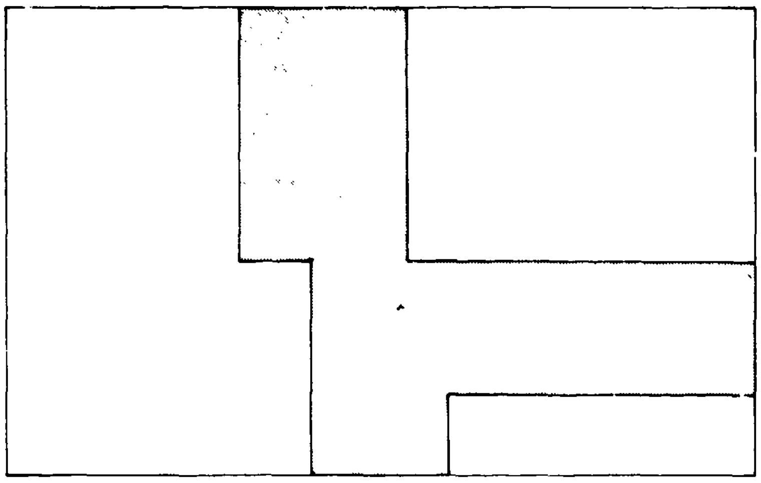


Figure 2.16. The region boundary map after the GLEC test.

region 3. Figure 2.17 shows the final segmentation results. Note that edges A, B, and C in Fig. 2.14 are the strong local line edges.

2.4.3 Selection of the Local Edge Operator

The majority of edges in an image can be detected by a gradient or modified gradient edge operator. The edge detection approach discussed in this section is based on the Sobel operator [3]. This particular operator has been shown to be quite useful in a number of comparison experiments involving a variety of edge detectors [35, 36].

2.4.4 Data Structure for Boundary Pixel Storage

The adjacent edges may be used in the coincidence test to determine how to merge the atomic regions. The data points of the adjacent edges can be stored by using a three-level hierarchical structure, based on the indexed address and the stored record address concepts to assure efficient use of memory. The atomic regions are first labelled by a sequence of numbers according to a top-bottom and left-right manner. The first level of the structure is a one-dimensional array. The location i of this array stores an index address which points to a location on the second level of the structure, which is a two-dimensional array. In this array, the region label of a region adjacent to region i is stored in the first row and the address of the corresponding boundary points of the adjacent edges is stored in the second row. If more than one region is adjacent to region i, the region labels of these adjacent regions and the addresses of the corresponding boundary points of the adjacent edges may be sorted and stored consecutively in this array. An end mark is inserted to separate the adjacent region labels and the

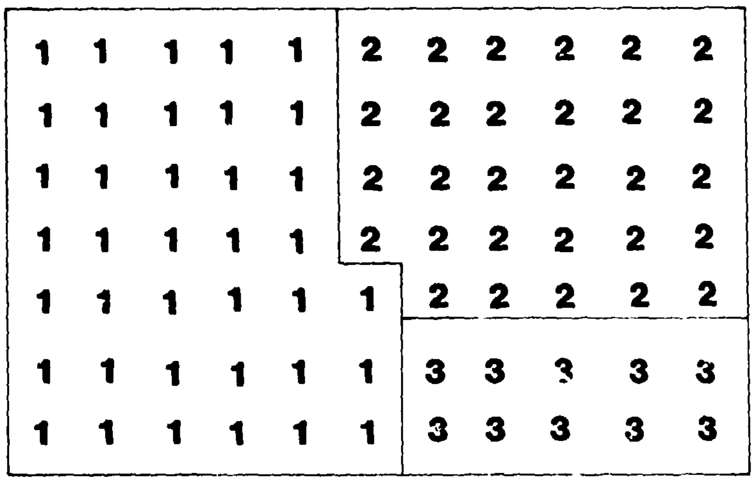


Figure 2.17. The final segmentation result.

addresses of the corresponding boundary points of two regions. The third level of the structure stores the boundary points of the adjacent edges which are addressed by the second level of the structure. Each boundary point set of adjacent edges is also separated by an end mark. The data structure is illustrated in Fig. 2.18. Region i is adjacent to region j and region k; the addresses of P_j and P_k are the locations of the boundary point set of the adjacent edges A_{ij} and A_{ik} at the third level of the data structure.

2.4.5 Tracing the Adjacent Edges of Regions

Four-connective adjacency may be used for classifying the interior pixels and the boundary pixels of a region. A pixel in a region is classified as an interior pixel if all of its four most adjacent neighbor pixels have the same region label as this pixel; otherwise, it is a boundary pixel. However, eight-connective adjacency can be used for tracing the adjacent edges between the regions. An eight-word buffer is used to store the region labels of the eight adjacent pixels of a boundary pixel. The region labels stored in the buffer indicate the region to which this boundary pixel is adjacent. Each boundary pixel may be adjacent to more than one region; however, only the boundary pixel of a single-pixel isolated region may be adjacent to eight different regions simultaneously. The boundary pixel and the adjacent region labels may be sorted and stored in the previously mentioned data structure.

2.4.6 Partitioning the Atomic Regions into Equivalence Classes

The boundary point set of the adjacent edges between two regions may easily be retrieved from the data base. If the adjacent edges of

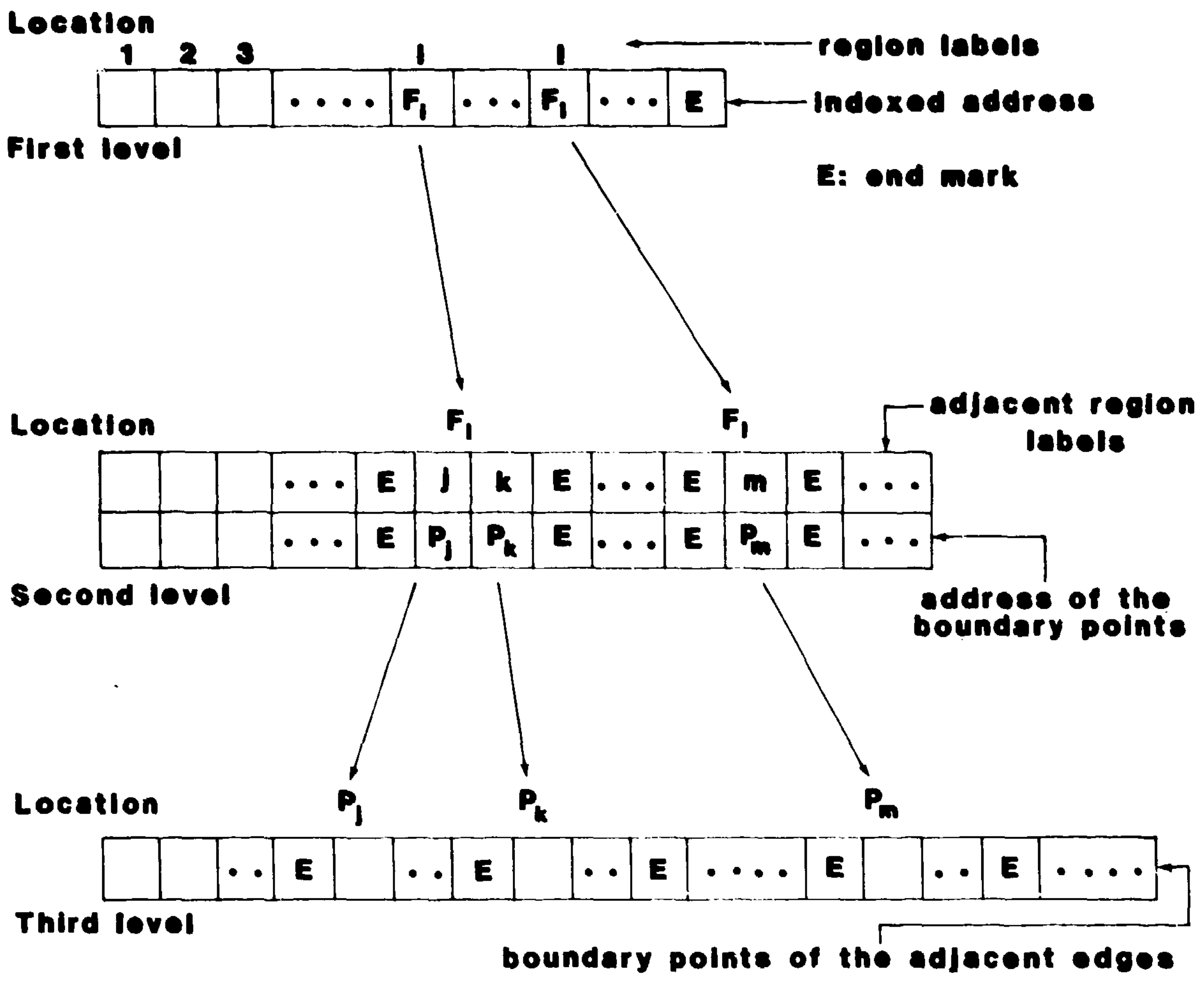


Figure 2.18. The three level hierarchical data structure for storing the boundary points of the adjacent edges.

these two regions satisfy any of the criteria specified in GLEC test, they are in the same equivalence class and need to be merged together. To partition the equivalence classes in a set is a problem of determining the transitive closure of a set [37]. In graph-theoretic terms, all the elements in an equivalence class which are strongly connected in a graph may be considered as a supernode and replaced by a proxy element [38]. In GLEC, this is equivalent to designating a new label to replace all the old atomic region labels in an equivalence class. An efficient algorithm is presented in this section to partition the atomic regions into equivalence classes and relabel these classes at the same time.

This algorithm uses a one-dimensional array for which location i stores the class label which indicates the equivalence class to which the region belongs. Assume that the class label of region R_i is L_i. Before the GLEC test, an i is stored in location i since all the atomic regions are in different classes; the region label is used as the class label. Since the adjacent edges are stored in a sequential linear list, the GLEC test will be sequentially processed region by region, starting from region one. Also, the adjacent edges of A_{uv} and A_{vu} are processed simultaneously in the GLEC test. The atomic region may be classified into the equivalence classes due to the transitive nature of an equivalence relation by the following decisions:

1. For $R_u \sim R_v$, if $u > v$, let $i = v$, $j = u$; otherwise $j = v$, $i = u$.
2. For $L_i < L_j$, and $L_j = j$, then $L_j = L_i$, go to 6.
3. If $L_j = L_i$, go to 6.
4. If $L_i < L_j$ and $L_j \neq j$, for all n let $L_n = L_i$ if $L_n = L_j$; go to 6.

5. If $L_i < L_j$ for all n let $L_n = L_j$ if $L_n = L_i$.

6. Next equivalence relation.

The final step is to reassign the class labels in the array to a sequence of numbers. A simple example may be used to illustrate these steps. Assume that the following sequence of relations are obtained through the GLEC test: $R_1 \sim R_2$, $R_1 \sim R_4$, $R_3 \sim R_8$, $R_5 \sim R_6$, $R_5 \sim R_9$, $R_6 \sim R_9$, $R_7 \sim R_8$, $R_8 \sim R_9$. Then, Fig. 2.19 shows the sequence of the classification results. There are two equivalence classes in this example: $R_1 \sim R_2 \sim R_4$ and $R_3 \sim R_5 \sim R_6 \sim R_7 \sim R_8 \sim R_9$.

2.4.7 Merging Small Atomic Regions Which are Coincident with Strong Local Edges

As previously described, the region boundary map may be produced by the coincidence test of the global and local edges. However, some of the small atomic regions may not be merged if the entire region or a great portion of this region is coincident with the strong edges. As a result, the resulting region boundary map will always contain the region boundary of these small atomic regions. These atomic regions may be considered "false" regions since they are at the same location as the strong edges. To merge these small regions, the following decision rule is used. If the overlapping area of a region and the strong edges is greater than a threshold, then this region will be merged to an adjacent region with the smallest contrast from it. The contrast of the two regions is defined as the absolute difference of the average gray level of two regions.

	1	2	3	4	5	6	7	8	9			Region Labels
Start	1	2	3	4	5	6	7	8	9	E	....	Class Labels
$R_1 \sim R_2$	1	1	3	4	5	6	7	8	9	E	....	Using step 2
$R_1 \sim R_4$	1	1	3	1	5	6	7	8	9	E	....	Using step 2
$R_3 \sim R_8$	1	1	3	1	5	6	7	3	9	E	....	Using step 2
$R_5 \sim R_6$	1	1	3	1	5	5	7	3	9	E	....	Using step 2
$R_5 \sim R_9$	1	1	3	1	5	5	7	3	5	E	....	Using step 2
$R_6 \sim R_9$	1	1	3	1	5	5	7	3	5	E	....	Using step 3
$R_7 \sim R_8$	1	1	3	1	5	5	3	3	5	E	....	Using step 5
$R_8 \sim R_9$	1	1	3	1	3	3	3	3	3	E	....	Using step 4
Reassign Class Label	1	1	2	1	2	2	2	2	2	E	....	Final Class Labels

Figure 2.19. An example of classifying the equivalence classes of regions.

2.4.8 Adjusting the Region Boundaries

The global edges and the local edges are assumed coincident to each other if either one matches the other in a four-connective manner. Therefore, the worst case of the result using the previous steps will be that some of the adjacent edges will be two pixels away from the real object boundaries (or the strong edges). A better match may be obtained by shifting and merging the boundaries between two regions. As shown in Fig. 2.20, let A_{uv} = {a,b,c,d} and A_{vu}={a',b',c',d'}. Then,the region boundaries of region u and region v will still be considered as coincident with the strong local edges, shown as the cross-hatched area of Fig. 2.20, since one of them (A_{vu}) is four-connective adjacent to the strong edges. However, the boundary points of A_{uv} are two pixels away from the strong edges, and those of A_{vu} are one pixel away. Therefore, it is appropriate to shift the boundaries of A_{uv} such that a better match of the global edges and the local edges may be obtained. In GLEC, a shift test is performed on the two related adjacent edges, i.e., A_{uv} and A_{vu} of region u and region v. This test results in a shift of one of the adjacent edges, A_{uv}, to improve the percentage of complete match of the global edges and the local edges. It is evident from Fig. 2.20 that A_{vu} will have a higher percentage of the one pixel away global-local match edges than A_{uv} of region u and region v. This test results in a shift of one of the adjacent edges, A_{uv}, to improve the percentage of complete match of the global edges and the local edges. It is noted in Fig. 2.20 that A_{vu} will have a higher percentage of the one pixel away global-local match edges than A_{uv}. The points of e, f, g, h and e', f', g', h', are the interior points of region v; therefore, if the initial boundary points

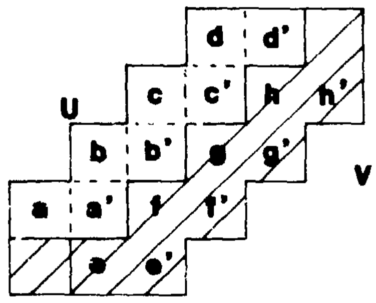


Figure 2.20. Shifting the adjacent edges of two regions will result in a better match of the global edges and the local edges.

of region v, i.e., points a', b', c', d', are merged to region u, this will result in a better match of the global edges and local edges. The same test is applied repeatedly until no further improvement can be made at these adjacent edges of region u and region v. The final adjacent edges of A_{uv} will contain the points of e, f, g, h, and A_{vu} will contain e', f', g', h'. By applying this method, the final region boundaries will have a better match with the strong local edges.

2.4.9 Examples of Segmentation Using the GLEC Technique

Four images are used in this section as examples for low-level object location using GLEC segmentation. These images include blocks, a building, aerial photographs, and a computed tomography image of a chest section.

The atomic regions for GLEC segmentation were generated by slicing the intensity histogram of the image into bands at its local minima, which were determined in the following way: The histogram was first slightly smoothed by an exponential filter in the spatial frequency domain [3]. The decay constant of this filter was set at 50 for all the segmentation examples in this section. The points with more than two consecutively increasing and decreasing values (i.e., up-up-down-down) were selected as the local maxima. The local minima were searched and set between the local maxima. The adjacent pixels were merged into an atomic region if the intensity level of these pixels corresponded to the same band in the intensity histogram. A region label was assigned to each atomic region using a region-growing technique [3]. This region label represents the subordination of a pixel to the atomic region.

The threshold for segmenting the local edges is scene-content dependent. It varies for different scenes. The histogram percentile chosen for segmenting the strong edges varied from 12% to 30%, depending on the complexity of the scene content. For segmenting these scenes, only a single pass of GLEC method was applied.

A simple scene consisting of a block, a cylinder and a triangular-block-section is shown in Fig. 2.21(a). After applying the mode method, the scene was segmented into atomic regions. Figure 2.21(b) shows the outer boundaries of the atomic regions. Note that there are many false regions segmented which need to be merged. The first step is to split and merge the pixels of the small size regions to the most adjacent region of these pixels. This split-and-merge step may be considered as defining the resolution of the GLEC segmentation. For instance, splitting and merging the atomic regions with size less than 3 pixels in Fig. 2.21(b), resulted in 183 such regions. Then these atomic regions were labelled according to the top-bottom and left-right sequence discussed previously. At this point, a region boundary map is required to start the GLEC segmentation. Figure 2.21(c) shows 14% of the highest intensity edges in the scene. Figure 2.21(d) shows the result of a binary correlation between 2.21(b) and 2.21(c). Then a coincidence test was conducted such that if the match of the adjacent atomic region boundaries and the edges in Fig. 2.21(d) was higher than 95%, the whole adjacent atomic region boundaries was retained; otherwise, they were eliminated. Figure 2.21(e) shows the "expected" boundary map of the three objects in this simple scene. Notice that if a lower percentage (<95%) were selected, the shadow of the block might be segmented as well. The atomic

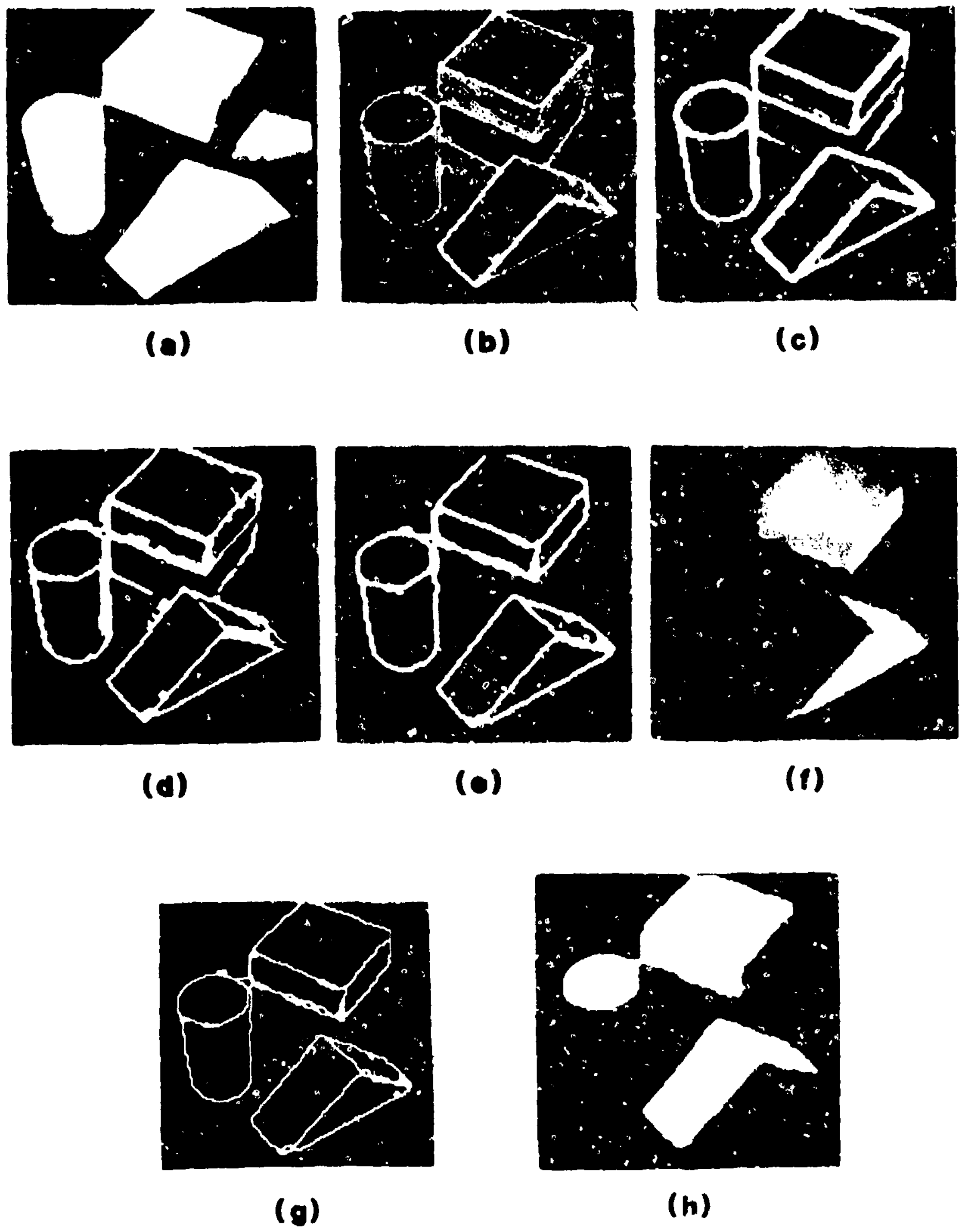


Figure 2.21. (a) Original image. (b) Outer atomic regions. (c) 14% of the highest intensity edges. (d) Binary correlation results of (b) and (c). (e) Region boundary map. (f) Segmentation results using the boundary map. (g) Outer region boundaries of (f). (h) Segmentation results after merging the small regions on the object border with the lowest contrast.

regions were then merged using procedure in Section 2.4.6 and the boundary map of Fig. 2.21(e). Figure 2.21(f) shows the result. Figure 2.21(g) shows the outer region boundaries of Fig. 2.21(f). Note that the small regions on the object borders cannot be merged no matter what threshold for the percentage of match is used since all the adjacent edges are completely coincident with the strong local edges. These small regions are most likely false regions since they are at the same location of the strong local edges. These regions should not be any object surfaces and need to be merged. These small regions were then merged to their adjacent regions with the lowest contrast using the procedure described in Section 2.4.7. Figure 2.21(h) shows the final segmentation result. One can find that the only distortion of this segmentation result is that a small portion of the shadow is merged with the lower corner of the block. All the objects in the scene have been segmented.

As another illustration, a scene consisting of a man-made object and natural objects was segmented. Figure 2.22(a) shows an image of a building, with trees and shrubs growing around the structure and along the sidewalk. The sign of the building is clearly seen. There is a shadow of the trees on the sidewalk. Figure 2.22(b) shows the outer region boundaries after this image was segmented by the mode method. Figure 2.22(c) shows the 30% of the highest intensity Sobel edges. Figure 2.22(d) shows the results after the GLEC segmentation. The threshold of percentage of match in the GLEC test was 75%. In Fig. 2.22(d), the windows, the door, the sign and the sidewalk have been clearly segmented. The shrubs beside the sidewalk have been merged with the building since no strong edges are between the building and the shrubs.

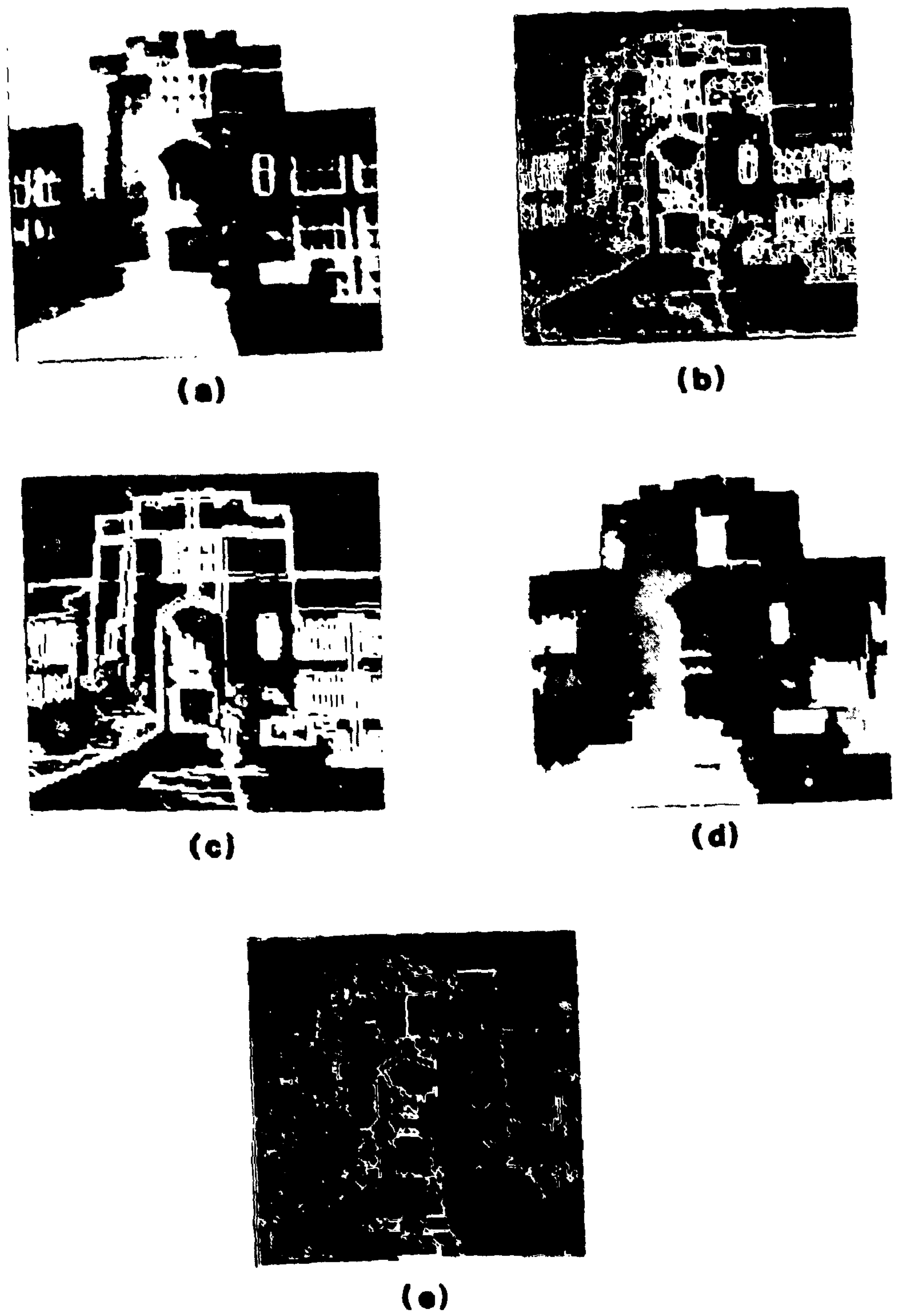


Figure 2.22. (a) Original. (b) Outer atomic regions. (c) 30% of the highest intensity edges. (d) Results of segmentation. (e) Outer region boundaries of (d).

Figure 2.22(e) shows the outer region boundaries of Fig. 2.22(d). Most objects in the scene are well-segmented.

GLEC segmentation may also be applied to edge linkage. The global edges or the region boundaries are used to link the local edges. Edge linking usually involves the steps of thinning the edges and determining the edge orientation by a set of compass masks [41]. Linking the edges using the global edges has several advantages. The edges which are GLEC edges are stable, strong edges. The edges have the same thickness since they are composed of the inner region boundaries. The structural relationship and the geometrical relationship of the edges are retained since all of them are connected. Any one section or many sections of edges between two junctions of the edges can be extracted since they are the adjacent edges of two adjacent regions. As an example, a portion of a strip of an aerial photograph is shown in Fig. 2.23(a). Figure 2.23(b) shows the outer region boundaries after they were segmented by the mode method. Figure 2.23(c) shows 12.5% of the highest intensity edges and Fig. 2.23(d) shows the final segmentation result using 65% as the percentage of match in the GLEC test. Comparing Fig. 2.23(c) and Fig. 2.23(d) one may easily find that most of the strong local edges are linked by using the global edges. Therefore, the threshold of the local edge intensity may be considered as a specification for the strong edges, and the threshold of the percentage of match can be a specification for edge linkage. Figure 2.23(e) shows all the inner region boundaries of Fig. 2.23(d). In this example, all the major regions have been well-segmented, except in the lower left corner of the upper section of the strip; a forest versus field region was merged due to the weak edge intensity at the region border. However, as shown in the following section, this region still can be segmented by using a sequential GLEC procedure.

2.4.10 Sequential GLEC Segmentation

In some cases, the global edges and the local edges are distributed as shown in Fig. 2.24. Assume that 60% is used as a criterion for defining a match of the global edges and the local edges in the GLEC test; then the following equivalence relation may be found from Fig. 2.24: $R_1 \sim R_2$, $R_2 \sim R_3$, $R_3 \sim R_4$. Since the relation is transitive, then all the regions will be in the same equivalence class. As a result, all of the regions will be merged together. However, if a sequential procedure is designed in such a way that the GLEC algorithm is applied iteratively according to a match percentile sequence, say, $T_{c1} = 10\%$, $T_{c2} = 60\%$, then three equivalence classes will be partitioned: R_1, $R_2 \sim R_3$, and R_4. It is obvious that the latter result is more reasonable than the former one.

Application of the sequential GLEC procedure to the upper section of the aerial photograph of Fig. 2.23(a) resulted in Fig. 2.25(b). Note that all major regions corresponding to different materials were segmented. Figure 2.25(c) shows the region boundaries, and Fig. 2.25(d) shows 20% of the highest intensity edges for the reference map. Since the adjacent boundary data of the segmented regions is recorded in GLEC segmentation, the edges may be selected by section from the image based on certain features along the location of the boundary points. For example, certain boundary sections are selected, as shown in Fig. 2.25(e), if the region contrast is higher than 32, and in Fig. 2.25(f) is higher than 58.

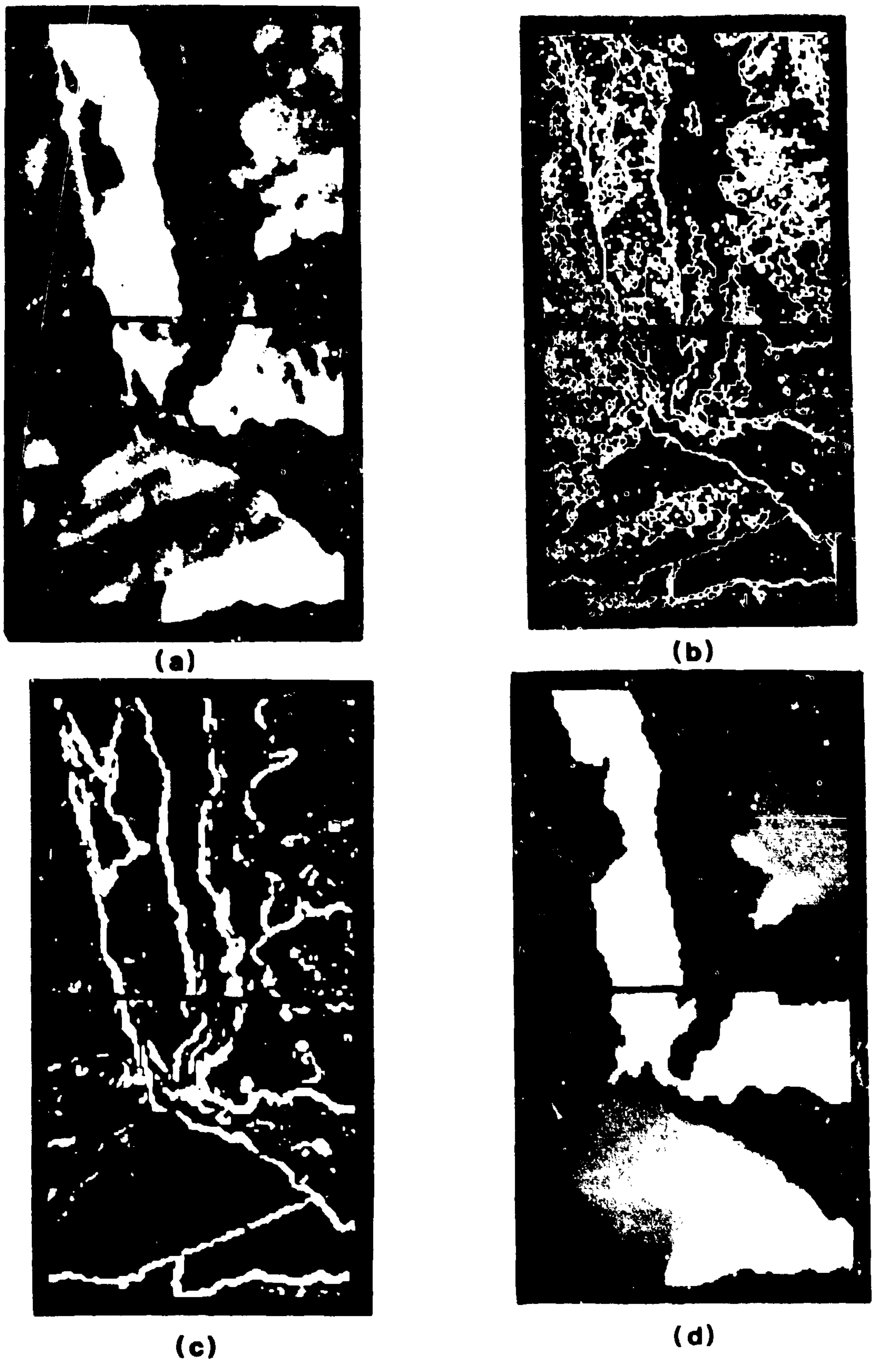
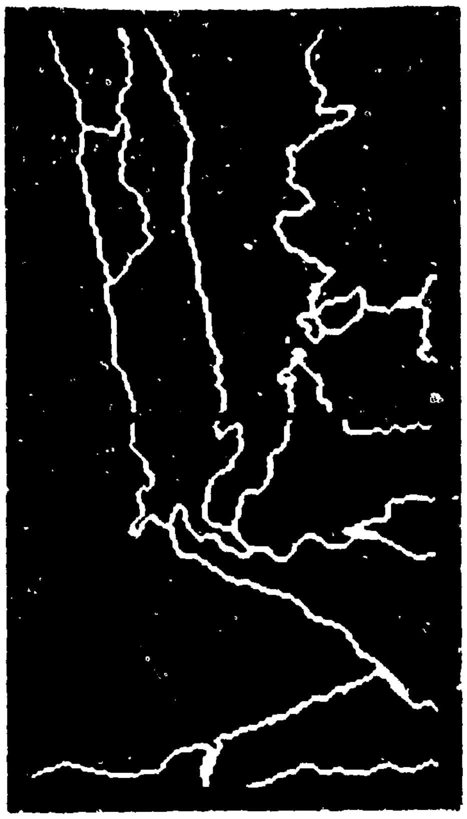

(a)
(b)
(c)
(d)

Figure 2.23. See next page for caption.



(e)

Figure 2.23. (a) Aerial image. (b) Outer atomic regions. (c) 12.5% of the highest intensity edges. (d) Segmentation results. (e) Outer region boundaries of (d).

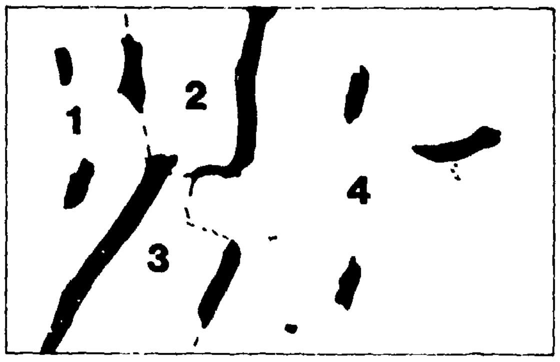


Figure 2.24. Example showing the need for sequential GLEC segmentation.

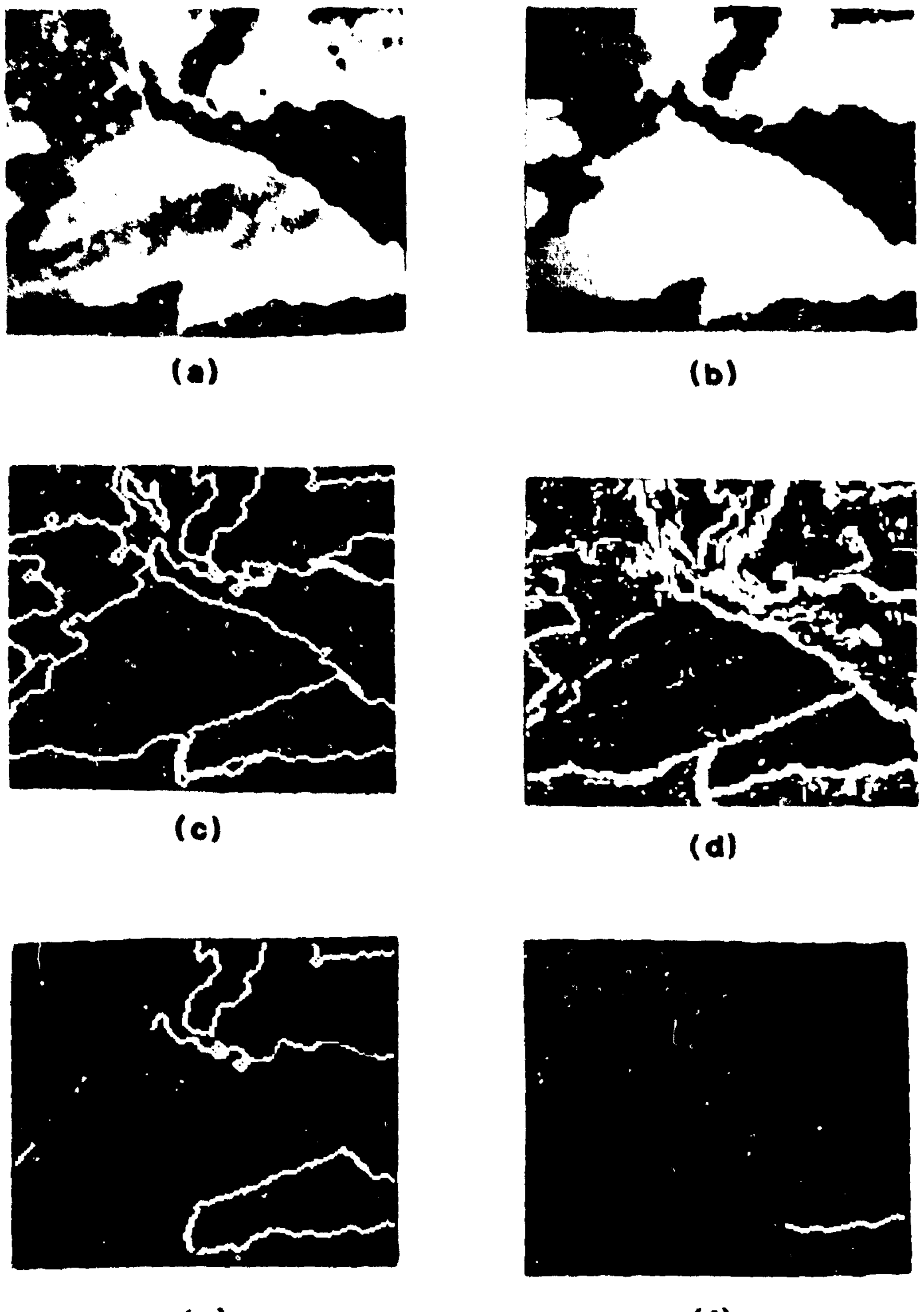


Figure 2.25. (a) Section of an aerial photograph. (b) GLEC segmentation. (c) Region boundaries of (b). (d) 20% of the highest intensity edges. (e) Selected edges with a contrast difference higher than 32. (f) Selected edges with a contrast difference higher than 58.

The region contrast is defined as the difference of the mean intensity level between two adjacent regions.

The above examples show that significantly improved results may be obtained by using the GLEC segmentation method. As with most segmentation methods, thresholds must also be used in the GLEC segmentation. Three thresholds are used in the method: a threshold for extracting the expected object borders from a local edge-processed image, a threshold for defining the percentage of connectivity of the strong local edges to be considered as valid object borders, and a threshold for merging the small regions which are at the same location as the strong edges.

The first threshold is defined by how strong the edges in a scene will be considered as the object borders. If the threshold is set too low, some of the object borders will be missed. If the threshold is set too high, many noisy local edges will also be extracted along with the object borders. Both cases will induce errors in GLEC segmentation. Usually, using a threshold corresponding to 10% up to 30% of highest intensity edges in the scene will be acceptable. In general, selecting a good threshold is scene-content dependent. A trial-and-error method may be required. However, selecting the threshold in a local edge-processed image is still an open problem in image processing.

The second threshold in GLEC defines the percentage of match in the GLEC test and is equivalent to defining the percentage of connectivity of the local edges to be considered as the valid object borders. Usually, selecting a threshold not lower than 65% will be acceptable.

The third threshold defines the percentage of overlapping area between a region and the strong edges which must be exceeded to be considered as a valid region. Usually, selecting a threshold lower than 20% will be acceptable. In other words, a region which overlaps more than 80% with the strong edges will be merged.

The atomic regions are generated by slicing the histogram at its valley. If the histogram of an image appears unimodal or uniformly distributed, an edge-preserving smoothing algorithm may be used as a pre-processor. The histogram of the smoothed images usually will appear multimodal [43, 44] and the GLEC algorithm will be applicable.

Edge information indicates a step change, and the step change of the object surfaces will normally produce edges. Some of the object surfaces may not be bounded by the intensity step edges, such as curved surfaces; however, the silhouette of the entire object may still show intensity step changes, and the object may still be segmented. Certain scene regions, such as trees on radar images, may be segmented by using texture changes, rather than intensity changes.

2.5 Recognition

Recognition is basically a labelling process; that is, the function of recognition algorithms is to identify each segmented object in a scene and to assign a label (e.g., road, vehicle, building) to that object. The design of recognition procedures for scene analysis consists of two basic steps: selection of a set of shape descriptors, and selection of a classification strategy.

2.5.1 Shape Descriptors

The principal descriptors used in scene analysis are based on shape and amplitude (e.g., intensity) information. Shape descriptors attempt to capture invariant geometrical properties of an object. This has been an illusive goal which has led to an impressive number of proposed techniques for shape description [45, 46]. Techniques for shape analysis and description are either global (region) or boundary oriented. Global techniques include principal axes analysis [3], texture [3, 47], two- and three-dimensional moment invariants [3, 48], geometrical descriptors such as perimeter squared/area and the extrema of a region [47, 49], and topological properties such as the Euler number [3], and decomposition into primary convex subsets [50]. Boundary techniques are based on thinned or skeletonized scene components [3, 47, 55]. They include: Fourier descriptors [3], chain codes [51, 52], graph representation (of which strings and trees are special cases) [50], and shape numbers [123, 124]. Variations and inconsistencies in shape are handled by the use of models [53], "rubber masks" [122], relaxation [54], and syntactic techniques [50].

Many shape analysis schemes are combinations of more than one of these approaches. For instance, syntactic methods often employ a set of productions in a pattern grammar to describe the global interconnection properties of shape features where the features themselves are primitive components obtained by decomposition. One of the most recent syntactic techniques [56] uses attributed shape grammars in which each nonterminal or terminal component has an attached vector of features. Other recently reported methods directly employ concepts of applied modern algebra to develop structural models of shapes [53]. In this section we illustrate procedures for shape analysis ranging from structural methods to more traditional techniques based on Fourier and invariant moment descriptors.

2.5.1.1 Structural Models of Shape

In this section we consider shape identification and matching technique based on finite sets and relations on set elements. The procedure requires that shapes of interest be described by discrete mathematical models against which shapes whose classifications are sought can be compared [53].

Formally, a shape description D = (N,T,G,P,R) consists of a shape name N; a type T; a set of global attributes G; a set of primitives P; and a set $R = \{R_1, \ldots, R_k\}$ of relations on P such that, for each k, $R_k \subseteq P^{N_k} \times L_k$ where N_k is a positive integer and L_k is a set of labels. The function of a shape description is to characterize a shape by its name, type, global characteristics such as the number of primitives, and labeled relations among primitives that embody the structures of the shape.

As an illustration, consider a block capital letter "E" as shown in Fig. 2.26 for which the primitives are the pixel clusters 1, 2, 3, and 4, and the intrusion clusters 11 and 12. The shape description suggested by Shapiro [53] is D = ('E', letter, $G,P\{R_1,R_2\}$) where

G = {(number of simple parts, 4), (number of intrusions, 2)},

P = {1,2,3,4,11,12},

R_1 = {(11,3,12,p), (12,3,11,p)},

R_2 = {(1,11,2,i), (1,11,3,i), (11,12,3,i), (1,12,4,i), (2,11,1,i), (3,11,1,i), (3,12,1,i), (4,12 1,i)}.

G gives some global information about the formation of letter 'E', and P contains the primitives. R_1 is a "protusion relation" in which the entry (11,3,12,p) indicates that a straight line can be drawn from a boundary point on 11 through 3 to a boundary point on 12, or, in words,

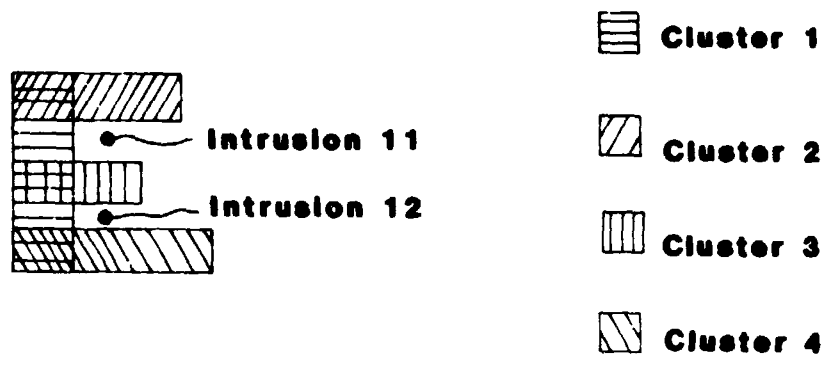


Figure 2.26. Simple parts and intrusions for prototype of capital letter 'E'.

that simple part 3 protrudes between 11 and 12. R_2 is an "intrusion relation" in which, for instance, the entry (1,11,2,i) indicates that a straight line can be drawn from a boundary point on 1 through 11 to a boundary point on 2, that is, that simple parts 1 and 2 touch or nearly touch and form part of the boundary of intrusion cluster 11. The information in R_1 and R_2 may be more concisely represented by deleting the entry labels, observing that symmetricity holds, and using the new relations $R_p = \{(11,3,12)\}$ and $R_i = \{(1,11,2), (1,11,3), (1,12,3), (1,12,4)\}$.

As is the case for many pattern recognition tasks, shape analysis in practice often requires that imperfectly formed or noisy candidate shapes be matched to perfect prototypes. The set of discrete shape descriptions can be used with various algorithms that allow for different labelings and relations in a candidate as compared with a prototype but for which the fundamental structure of the shape is preserved.

For this purpose, a relational homomorphism is defined as follows. Let $R \subseteq A^N \times L$ and $R' \subseteq B^N \times L$ be relations that attach labels to n-tuples of elements from sets A and B respectively, and let $h:A \rightarrow B$ be a mapping from A to B. Then h is a relational homomorphism if $R \circ h \subseteq R'$ where $R \circ h = \{(h(a_1),h(a_2), \ldots, h(a_N)) \mid (a_1,a_2, \ldots, a_N,) \text{ in } R\}$. Two shapes represented by labeled relations are said to match if a relational homomorphism can be found from one representation to the other.

As an illustration, let Fig. 2.26 together with relations R_p and R_i defined above be the prototype shape description for the capital 'E'. Suppose that Fig. 2.27(a) is a candidate shape and that the simple part and intrusion clusters shown in Fig. 2.27(b) are obtained by preprocessing.

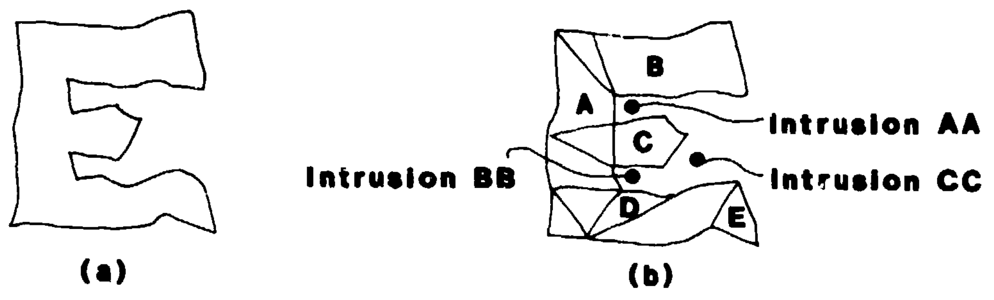


Figure 2.27. Candidate capital letter with simple parts and intrusions.

The compact representations of the intrusion and protrusion relations with symmetric entries omitted are as follows for Fig. 2.27:

$$R_{p2} = \{(AA,C,BB),(AA,C,CC)\},$$
$$R_{i2} = \{(A,AA,B),(A,AA,C),(A,BB,C),(A,BB,D),(A,BB,E),$$
$$(D,BB,E),(D,CC,E)\}.$$

For the candidate to match the prototype, there must be a relational homomorphism $h:\{1,2,3,4,11,12\}\to\{A,B,C,D,AA,BB,CC\}$. At least four such mappings can be found and appear in Table 2.1. For h^1, for instance, we have

$$\begin{aligned} R_p o h^1 &= \{(AA,C,BB)\} \\ &= \{(h^1(11),h^1(3),h^1(12))\} \\ &\subset R_{p2} \end{aligned}$$

and

$$\begin{aligned} R_i o h^1 &= \{(A,AA,B),(A,AA,C),(A,BB,C),(A,BB,D)\} \\ &= \{(h^1(1),h^1(11),h^1(2)),(h^1(1),h^1(11),h^1(3)), \\ &\quad (h^1(1),h^1(12),h^1(3)),(h^1(1),h^1(12),h^1(4))\} \\ &\subset R_{i2}. \end{aligned}$$

Note that only containment of $R_p o h^1$ in R_{p2} and of $R_i o h^1$ in R_{i2} is required to establish a match. Homomorphisms h^3 and h^4 establish a mirror image type of match of candidate and prototype.

The developer of the prototype shape descriptions might deem various entries in the relations more important than others and might require that even a misshapen candidate match to at least a minimal degree. Let $w:R\to[0,1]$ assign weights in the interval [0,1] to entries in prototype relation R such that

X	$h^1(X)$	$h^2(X)$	$h^3(X)$	$h^4(X)$
1	A	A	A	A
2	B	B	D	E
3	C	C	C	C
4	D	E	B	B
11	AA	AA	BB	BB
12	BB	BB	AA	AA

Table 2.1. Four possible mappings.

$$\sum_{r \text{ in } R} w(r) = 1 \qquad (2.5\text{-}1)$$

then w is a <u>weighting function</u> for structure relation R. If h is a relational homomorphism from a prototype to a candidate shape, then the candidate is <u>within ε</u> of the prototype if

$$\sum_{\substack{r \text{ in } R \\ r \text{ not satisfied by } h}} w(r) \leq \varepsilon \qquad (2.5\text{-}2)$$

In other words, the sum of the weights of prototype relation entries for which h does not establish a correspondence with candidate relation entries must not exceed ε for a successful match to be declared. By controlling the weights and ε, the designer can control the quality with which shapes are identified.

2.5.1.2 Shape Description Using Attributed Grammars

One of the major criticisms of the syntactic approach to pattern recognition [57, 58] is that the process of extracting the primitives of a pattern is separated from the process of structural analysis using a grammar; that is, the act of scanning a pattern to identify its primitive components usually is entirely completed before the attempt to parse the pattern representation via a pattern class grammar. The procedure described in this section is based on attributed grammars in which primitive extraction and syntax analysis can occur simultaneously [56].

This method uses semantic and syntactic information given respectively by attributes of grammar symbols and by productions. An <u>attributed grammar</u> can be considered to be a conventional context-free string grammar (see Section 2.5.2) to whose nonterminals and terminals value attributes are attached. A parse of a string by an attributed grammar can be successfully completed if (i) the string is syntactically correct according to the productions, and (ii) the values of the symbols in the string are acceptable according to the grammar's rules for attributes. One can view the parsing of a recognizable string as a bottom-up construction of the string's derivation tree beginning with the terminals and their attributes; on a step by step basis, in order to add another level upwards in the tree, not only must a collection of tree nodes be mergeable to a single nonterminal, but the value attributes of those nodes must also generate an acceptable value attribute for the new nonterminal. (The attributes are sometimes called "action symbols" because an attributed grammar can be considered a <u>syntax-directed translation schema</u> [58] by which an acceptable input string is translated into an output action string to be executed on a computer.)

Shapes composed of curved line segments are treated as a line segment primitive with four attributes. For segment $p = X_1X_2$ from starting point X_1 to ending point X_2, with length L and differential element $d\ell$, these attributes are as follows:

$\vec{C} = \overrightarrow{X_1X_2}$, the vector locator;

$L = \int_0^L d\ell$, the total length of the curve;

$A = \int_0^L f(\ell)d\ell$, the total angular change along the curve; and

$S = \int_0^L (\int_0^S f(\ell)d\ell - \frac{A}{2}) ds$, the symmetry measure

A and S are computed using

$$f(\ell) = \lim_{\Delta\ell \to 0} \left[\begin{array}{l} \text{angle between the tanget lines to the curve} \\ \text{segment at } \ell - \frac{\Delta\ell}{2} \text{ and } \ell + \frac{\Delta\ell}{2} \end{array} \right]$$

We write $D(p) = (\vec{C},L,A,S)$ as the attribute values for curve segment p.

An angle primitive a is used to describe the connection angle between two curves. Its attribute, $D(a) = A$, is the angular change at the concatenation point of the two line segment primitives.

As an illustration, we take $p = X_1X_2$ in Fig. 2.28 to be composed of $p_1 = X_1X_3$ and $p_2 = X_3X_2$. If this is part of a shape boundary, there will be a production $p \to p_1\ a\ p_2$ in the shape grammar; and, in order to use this production in a parse, the attributes $D(p_1) = (\vec{C}_1,L_1,A_1,S_1)$, $D(p_2) = (\vec{C}_2,L_2,A_2,S_2)$, and D(a) would have to yield attributes $D(p) = (\vec{C},L,A,S)$ that are acceptable for the overall shape requirements. For a production of this form, there is an additivity property for synthesizing the attributes of p from those of p_1 and p_2; namely,

$$\vec{C} = \vec{C}_1 + \vec{C}_2 \quad \text{(in vector addition)},$$
$$L = L_1 + L_2,$$
$$A = A_1 + a + A_2,$$
$$S = S_1 + S_2 + 1/2[(A_1 + a)L_2 - (A_2 + a)L_1].$$

This property is associative.

A shape is represented as a string of curve and angle primitives that define its boundary. An attributed shape grammar for a shape labeled ℓ is $G_\ell = (V_\ell, T_\ell, P_\ell, S_\ell)$ where V_ℓ is the collection of nonterminals, including starting symbol S_ℓ; T_ℓ is the set of curve and angle primitives; the productions in P_ℓ have the form $S_\ell \to (XA)^jXA$ and $N \to (XA)^jX$, $N \neq S_\ell$, for X being a single nonterminal or curve primitive, A an angle primitive,

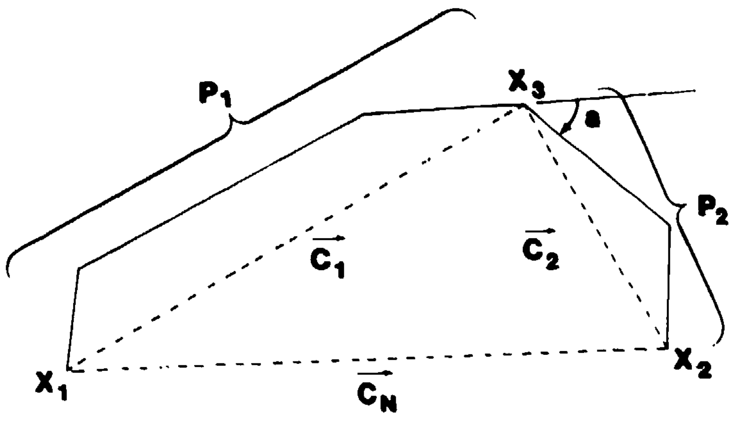


Figure 2.28. Curve primitive X_1X_2 decomposed into curve primitives X_1X_3 and X_3X_2 and angle primitive a.

and $(XA)^j$ denoting a sequence of j occurences of XA; and the synthesized attributes attached to the nonterminal on the lefthand side of a production are computed as described above. Figure 2.29 gives two shapes for a BAC111 airplane, and Fig. 2.30 gives a corresponding attributed shape grammar.

2.5.1.3 Shape Description Using Fourier Coefficients

Fourier tranform techniques may be used for shape description and classification in the following way. The boundary of a candidate shape is determined and its Fourier series representation as a function of two variables, x and y, is computed; after frequency domain computations to normalize the position, size, orientation, and starting point, the Fourier descriptors are compared with those of standard, prototype shapes and the best correspondence is selected as the classification of candidates [3, 59].

As a practical aspect of the computation, the discrete Fourier transform (DFT) is used for the $x - y$ coordinates of a finite number of boundary points taken as complex numbers $x + jy$ and obtained by uniformly spaced sampling. To normalize for size, one can multiply all Fourier coefficients by that factor which makes the first coefficient F(1) equal to 1. To normalize for orientation and starting point, one can adjust the phase angles by a factor $j\phi$ to produce standard phases for F(1) and the next two largest coefficients. An advantage of Fourier descriptors is that handling these aspects is much more convenient in the frequency domain than in the space domain.

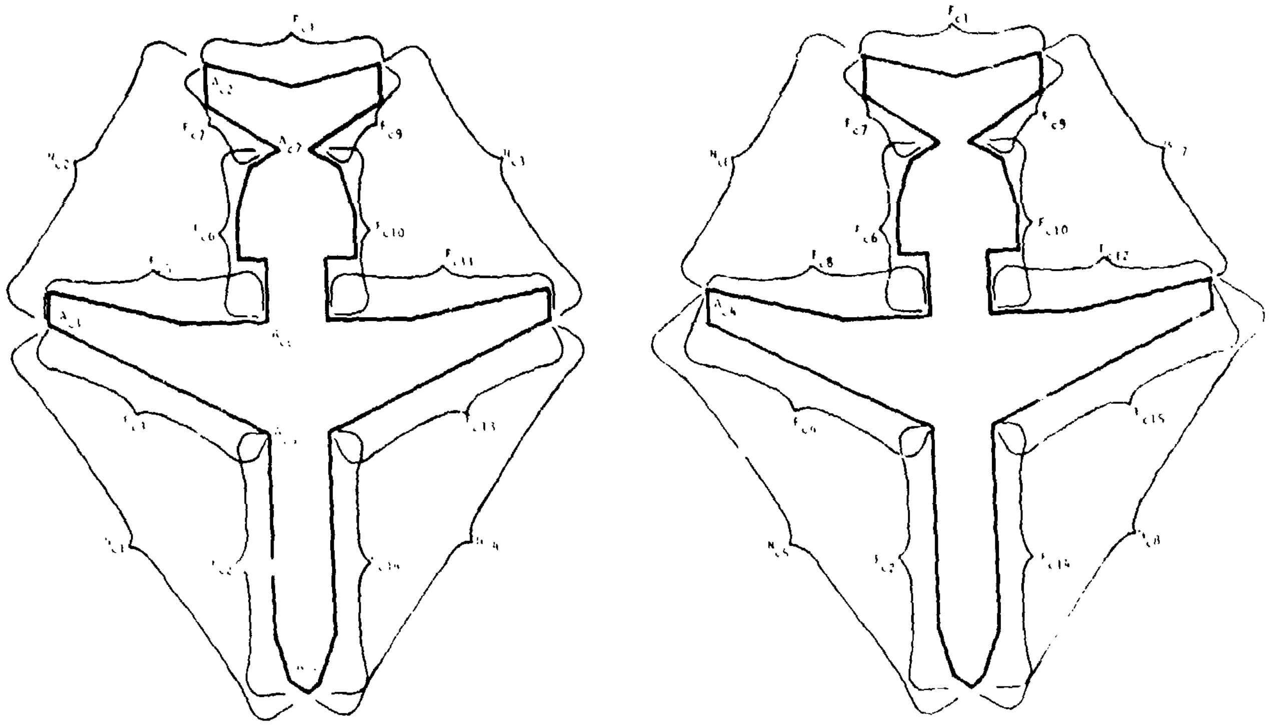


Figure 2.29. Two shapes using curve and angle primitives for the BAC 111 airplane.

$V_c = \{S_c, N_{ci} \mid 1 \le i \le 8\}$

$T_c = \{F_{cj}, A_{ck} \mid 1 \le j \le 15, 1 \le k \le 7\}$

$\mathbf{P_c}$:
$S_c \rightarrow N_{c1} A_{c1} N_{c2} A_{c2} F_{c1} A_{c2} N_{c3} A_{c1} N_{c4} A_{c3}$
$S_c \rightarrow N_{c2} A_{c2} F_{c1} A_{c2} N_{c3} A_{c1} N_{c4} A_{c3} N_{c1} A_{c1}$
$S_c \rightarrow F_{c1} A_{c2} N_{c3} A_{c1} N_{c4} A_{c3} N_{c1} A_{c1} N_{c2} A_{c}$.
$S_c \rightarrow N_{c3} A_{c1} N_{c4} A_{c3} N_{c1} A_{c1} N_{c2} A_{c2} F_{c1} A_{c2}$
$S_c \rightarrow N_{c4} A_{c3} N_{c1} A_{c1} N_{c2} A_{c2} F_{c1} A_{c2} N_{c3} A_{c1}$
$S_c \rightarrow N_{c6} A_{c2} F_{c1} A_{c2} N_{c7} A_{c4} N_{c8} A_{c3} N_{c5} A_{c4}$
$S_c \rightarrow N_{c8} A_{c3} N_{c5} A_{c4} N_{c6} A_{c2} F_{c1} A_{c2} N_{c7} A_{c4}$
$N_{c1} \rightarrow F_{c2} A_{c5} F_{c3}$
$N_{c5} \rightarrow F_{c2} A_{c5} F_{c4}$
$N_{c2} \rightarrow F_{c5} A_{c6} F_{c6} A_{c7} F_{c7}$
$N_{c6} \rightarrow F_{c8} A_{c6} F_{c6} A_{c7} F_{c7}$
$N_{c3} \rightarrow F_{c9} A_{c7} F_{c10} A_{c6} F_{c11}$
$N_{c7} \rightarrow F_{c9} A_{c7} F_{c10} A_{c6} F_{c12}$
$N_{c4} \rightarrow F_{c13} A_{c5} F_{c14}$
$N_{c8} \rightarrow F_{c15} A_{c5} F_{c14}$.

Figure 2.30. Attributed shape grammar for BAC 111 shapes.

The question of how many DFT coefficients must be computed has to be answered for each specific application. From the standpoint of Euclidean analysis, an infinite Fourier series represents a function F in area R by convergence in the mean; that is, letting F_k represent the subseries consisting of the first k terms of the full Fourier series for F, we have

$$\lim_{k\to\infty} \| F_k - F \| = 0 \tag{2.5-3}$$

where the norm $\| \cdot \|$ is the mean-square measure.

$$\| F_k - F \| = \Big(\iint_{\text{areaR}} [f_k(x,y) - F(x,y)]^2 dxdy\Big),^{1/2} \tag{2.5-4}$$

or its discrete summation analog for the DFT.

Since the infinite expansion is infeasible, a finite-dimensional subspace must be chosen and should be as small as possible consistent with the shapes to be dealt with. (Relatively few low frequency coefficients can characterize shapes that are grossly different, but the differences between shapes that vary only in small details require the information contained in higher frequency components.) The first K descriptors, say F_K, define the projection of F onto a finite dimensional subspace; and $D = F - F_K$ is the vector perpendicular to F's projection, so that $\| D \|$ is the distance from F to the subspace. To use this method, one must therefore set K large enough to insure that $\| D \|$ is small enough for each possible candidate shape to yield an identifiable representation of the original F.

An illustration of the use of Fourier descriptors for shapes is provided in Fig. 2.31. These four aircraft silhouttes were obtained in the following way:

(1) The normalized discrete Fourier descriptors were computed for 512 points on the original silhouttes' boundaries.

(2) The 32 lowest frequency terms were retained and all others were discarded.

(3) The inverse transforms of the modified 512-arrays were computed and plotted.

The projections of the original silhoutte functions onto the subspace spanned by 32 basis vectors in the Fourier series retain sufficient information to give reasonable approximations for the classification of these shapes. For line patterns not closed and not overlapping, as exemplified by the numeral in Fig. 2.32(a), the shape should be traced once, then retraced to form a closed boundary curve as in Fig. 2.32(b) where the normalize starting point is at the endpoint of the numeral [60].

2.5.1.4 Shape Approximation by Moments

The method of describing the shape of an image or subimage by computing moments is a scalar transformation technique. One of the advantages of this method is that simple linear combinations of moments are invariant under a number of similarity transforms, including translation, rotation, and scale change [3, 48, 59, 61].

Let $g(x,y)$ be the <u>characteristic function</u> of a shape in two dimensions, that is, let $g(x,y)$ be 1 for points within the shape boundary and 0 for points outside. The (p,q)th moment of the function is

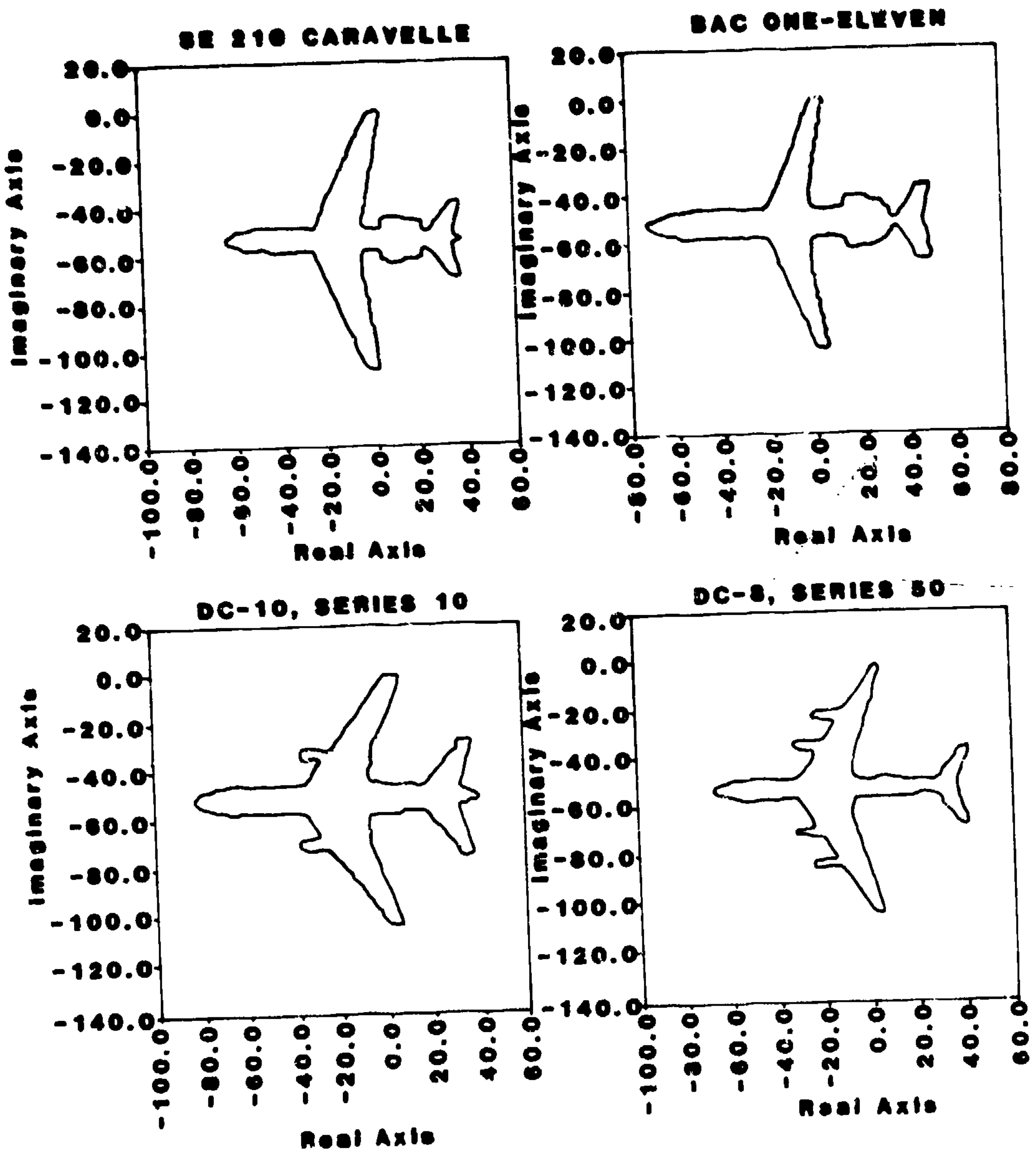


Figure 2.31. Shapes of four aircraft obtained by inverse Fourier transforms with 32 Fourier descriptors.

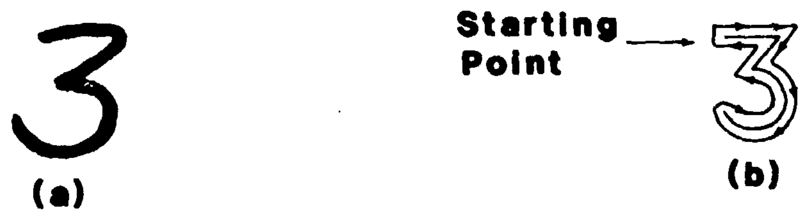


Figure 2.32. Numeral and its tracing to form a closed boundary.

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q g(x,y) dx\, dy \qquad (2.5\text{-}5)$$

for p,q = 0, 1, 2, ... A uniqueness theorem [62] states that if g(x,y) is piecewise continuous with a finite number of discontinuities and non-zero values in the x-y plane, the moments of all finite orders exist and there is a one-to-one correspondence between the set of moments $\{m_{pq} | p,q \geq 0\}$ and g(x,y). In fact, the moment generating function of g(x,y),

$$M(u,v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp(ux + vy)\, g(x,y)\, dx\, dy \qquad (2.5\text{-}6)$$

has a representation as a power series

$$M(u,v) = \sum_{p=0}^{\infty} \sum_{q=0}^{\infty} m_{pq} \frac{u^p}{p!} \frac{v^q}{q!} \qquad (2.5\text{-}7)$$

the coefficients of which are exactly the moments of the shape characteristic function [59].

The zeroth order moment m_{00} is the area within the boundary. m_{10} and m_{01} have the property that

$$\bar{x} = \frac{m_{10}}{m_{00}}, \quad \bar{y} = \frac{m_{01}}{m_{00}} \qquad (2.5\text{-}8)$$

are the coordinates of the shape's centroid. In general, the (p,q)th order central moment of g(x,y) is

$$\mu_{pg} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})^p (y - \bar{y})^q g(x,y)\, dx\, dy \qquad (2.5\text{-}9)$$

which, for a digitized image, is computed discretely as

$$\mu_{pg} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q g(x,y) \qquad (2.5\text{-}10)$$

The normalized central moments, denoted by η_{pq}, are computed as

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}{}^{\gamma}} \qquad (2.5\text{-}11)$$

for $\gamma = 1/2(p + q)$, $p + q = 2, 3, \ldots$. From the η_{qp}'s, a set of seven values invariant to translation, rotation, and scale change can be computed as follows:

$$\begin{aligned}
\psi_1 &= \eta_{20} + \eta_{02} \\
\psi_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
\psi_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} + \eta_{03})^2 \\
\psi_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2
\end{aligned} \qquad (2.5\text{-}12)$$

$$\begin{aligned}
\psi_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\
&\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
\psi_6 &= (\eta_{20} - \eta_{02})[\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
&\quad + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
\psi_7 &= (3\eta_{12} = \eta_{30})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\
&\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
\end{aligned} \qquad (2.5\text{-}13)$$

An illustration of these moment invariants is given in Fig. 2.33. The image in Fig. 2.33(a) is reduced to half size in 2.33(b), mirror-imaged in 2.33(c), rotated by 2 degrees in 2.33(d), and rotated by 45 degrees in 2.33(e). Table 2.2 lists the corresponding values of ψ_1 through ψ_7. There is reasonable agreement with the values for the original image, and the major source of error is said to be the digitized approximations of continuously-valued functions [3].

2.5.2 Classification Strategies

Classification strategies may be subdivided into two principal categories: decision-theoretic and syntactic [58, 74]. Decision-theoretic techniques are based on the use of decision (discriminant) functions. Let $\underline{x} = (x_1, x_2, \ldots, x_n)^T$ represent a column <u>pattern vector</u> with real components, where x_i is the ith descriptor measured on a given pattern. Since the components of $\underline{x}$ are real-valued variables, it is evident that each component of this vector represents a descriptor that can be expressed in terms of a scalar quantity (e.g., area). Given M pattern vector classes, ω_1, $\omega_2, \ldots, \omega_M$, the decision-theoretic approach consists of identifying M <u>decision functions</u> $d_1(\underline{x}), d_2(\underline{x}), \ldots d_M(x)$ with the property that, for any pattern $\underline{x}^*$ from class ω_i, $d_i(\underline{x}^*) > d_j(\underline{x}^*)$, $j = 1,2,\ldots,M, j \neq i$. The objective is to find M decision functions such that this condition holds for all classes with minimum error in misclassification.

Central to the decision-theoretic approach to pattern recognition is the concept of <u>learning</u>. Suppose that decision functions are expressed in the form $d_i(\underline{x}) = \sum_{k=1}^{K} w_{ik}\phi_k(\underline{x})$, where the w's are coefficients and the ϕ's are known functions of $\underline{x}$ (e.g., polynomial functions). Then, learning

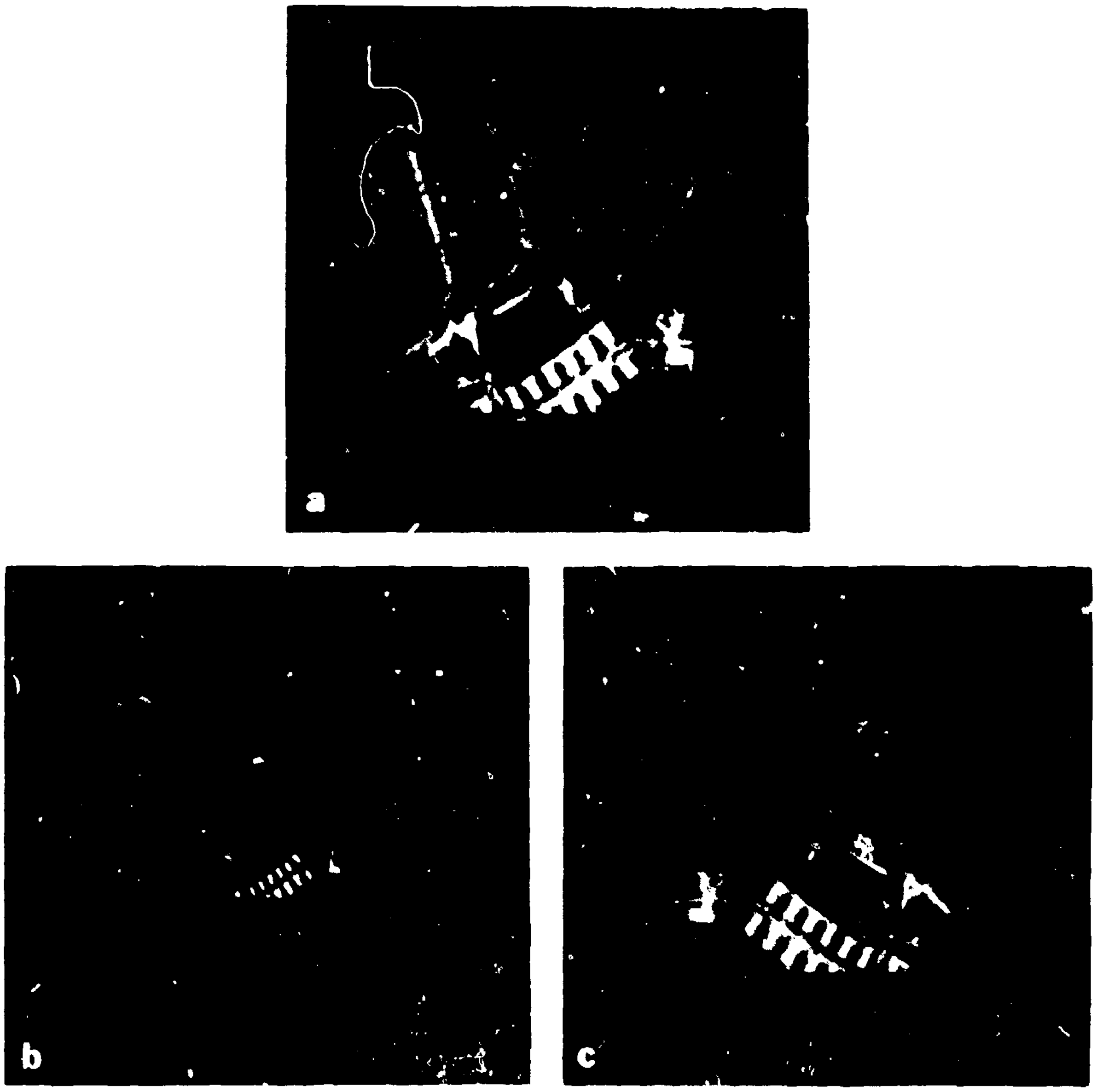


Figure 2.33. (a) Original image, (b) half size, (c) mirror image.

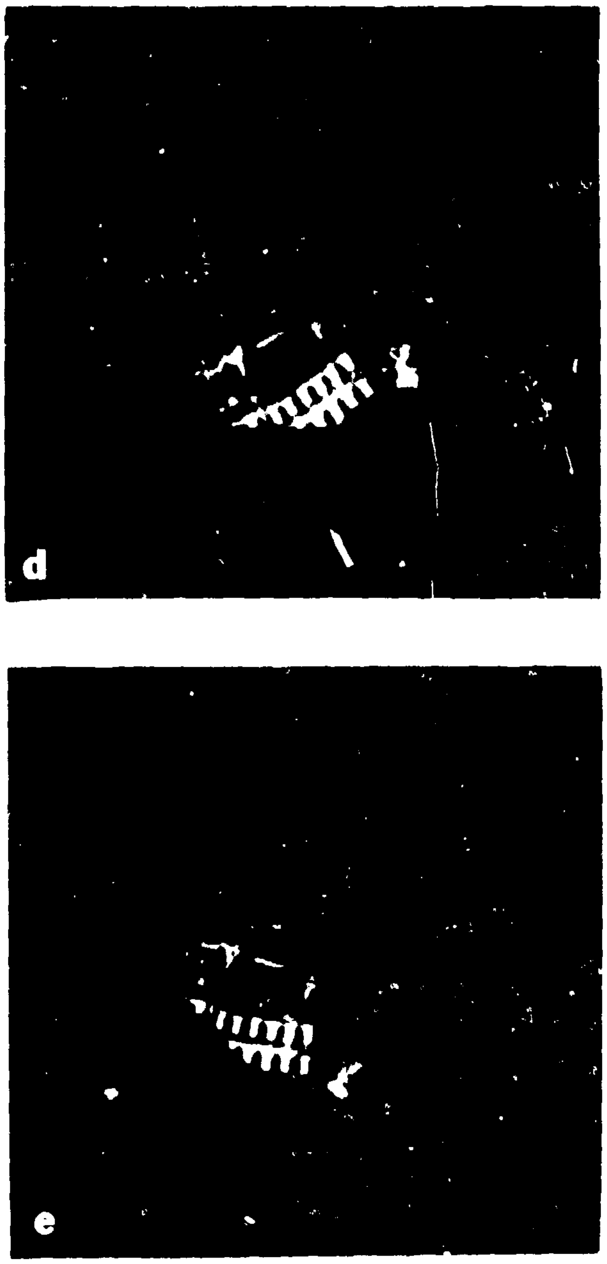


Figure 2.33. Continued. (d) Rotated 2 degrees, (e) rotated 45 degrees.

Invariant (log)	Original	Half size	Mirrored	Rotated(2°)	Rotated(45°)
φ_1	6.249	6.226	6.919	6.253	6.318
φ_2	17.180	16.954	19.955	17.270	16.803
φ_3	22.655	23.531	26.689	22.836	19.724
φ_4	22.919	24.236	26.901	23.130	20.437
φ_5	45.749	48.349	53.724	46.136	40.525
φ_6	31.830	32.916	37.134	32.068	29.315
φ_7	45.589	48.343	53.590	46.017	40.470

Table 2.2. Moment invariants for original and processed images.

in this context refers to the use of training patterns (i.e., pattern vectors whose class identity is known) to obtain the coefficients of the decision functions via the utilization of training algorithms. When the class identity of the pattern vectors is not known, the learning is said to be unsupervised.

Given M decision functions of the form described above, it is noted that values of $\underline{x}$ for which $d_i(\underline{x}) = d_j(\underline{x})$ (or $\sum_{k=1}^{k} w_{ik}\phi_k(\underline{x}) - \sum_{k=1}^{K} w_{jk}\phi_k(\underline{x}) = 0$) describe a hypersurface which partitions classes ω_i and ω_j. Thus, the decision-theoretic approach to pattern recognition may be viewed as the set of procedures which partition the pattern space by means of hypersurfaces. An optimal procedure is one that yields the lowest recognition error in achieving this partition.

Syntactic methods are based on the use of concepts from formal language theory [58]. The basic difference between decision-theoretic and syntactic pattern recognition is that the latter explicitly utilizes the structure of patterns in the recognition process. Decision-theoretic approaches, on the other hand, deal with patterns on a strictly quantitative basis, thus largely ignoring interrelationships between the components of a pattern. Of course, the existence of a recognizable "structure" is essential for the success of the syntactic approach. For this reason, syntactic pattern recognition research has been largely confined thus far to pictorial patterns which are characterized by recognizable shapes, such as characters, chromosomes, and particle collision photographs.

As a way of introduction to this area, consider the following definitions: An alphabet is any finite set of symbols. A sentence over an alphabet is any string of finite length composed of symbols from the alphabet. For example, given the alphabet {0,1}, the following are valid sentences: {0,1,00,01,10,...}. The terms string and word are also commonly used to denote a sentence. The sentence with no symbols is called the empty sentence, λ. For any alphabet V, we use V^* to denote the set of all sentences composed of symbols from V, including the empty sentence. The symbol V^+ denotes the set of sentences $V^* - \lambda$. A language is any set (not necessarily finite) of sentences over an alphabet.

As is true in natural languages, a serious study of formal language theory must be focused on grammars and their properties. A grammar is defined as the fourtuple:

$$G = (N,\Sigma,P,S) \tag{2.5-14}$$

where

N is a set of nonterminals (variables)

Σ is a set of terminals (constants);

P is a set of productions or rewriting rules;

S is the start or root symbol.

It is assumed that S belongs to the set N and that N and Σ are disjoint sets. The alphabet V is the union of sets N and Σ.

The language generated by G, denoted by L(G), is the set of strings which satisfy two conditions: (1) each string is composed only of terminals (i.e., each string is a terminal sentence), and (2) each string can be derived from S by suitable applications of productions from the set P.

The following notation is commonly used in this field. Nonterminals are denoted by capital letters: S, A, B, C, ... Lower-case letters at the beginning of the alphabet are used for terminals a, b, c, ... Strings of terminals are denoted by lower-case letters toward the end of the alphabet: v, w, x, ... Strings of mixed terminals and nonterminals are represented by lower-case Greek letters: $\alpha, \beta, \gamma, \delta, \ldots$.

The set P of productions consists of expressions of the form $\alpha \rightarrow \beta$, where α is a string in V^+ and β is a string in V^*. In other words, the symbol $\rightarrow$ indicates replacement of the string α by the string β. The symbol $\underset{G}{\Rightarrow}$ is used to indicate operations of the form $\gamma\alpha\delta \underset{G}{\Rightarrow} \gamma\beta\delta$ in the grammar G, that is, $\underset{G}{\Rightarrow}$ indicates the replacement of α by β by means of the production $\alpha \rightarrow \beta$, γ and δ being left unchanged. It is customary to drop the G and simply use the symbol $\Rightarrow$ when it is clear which grammar is being considered.

Grammars are classified according to the type of productions allowed in P. An <u>unrestricted grammar</u> has productions of the form $\alpha \rightarrow \beta$, where α is a string in V^+ and β is a string in V^*. A <u>context-sensitive</u> grammar has productions of the form $\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$, where α_1 and α_2 are in V^*, β is in V^+ and A is in N. This grammar allows replacement of the nonterminal A by the string β only when A appears in the context $\alpha_1 A \alpha_2$ of strings α_1 and α_2. An alternative definition is $\alpha \rightarrow \beta$, with both α and β in V^+ and the length of α not exceeding that of β. A <u>context-free grammar</u> has productions of the form $A \rightarrow \beta$, where A is in N and β is in V^+. The name context free arises from the fact that the variable A may be replaced by string β regardless of the context in which A appears. Finally, a regular grammar is one with productions of the form $A \rightarrow aB$ or $A \rightarrow a$, where A and B are variables in N and a is a terminal in Σ.

The concepts just discussed can be related to pattern recognition in the following manner. Suppose that we have two pattern classes ω_1 and ω_2. Let the patterns of these classes be composed of features from some finite set. We call the features terminals and denote the set of terminals by Σ. The term primitives is also so often used in syntactic pattern recognition terminology to denote terminals. Each pattern may be considered as a string or sentence since it is composed of terminals from the set Σ. Assume that there exists a grammar G with the property that the language it generates consists of sentences (patterns) which belong exclusively to one of the pattern classes, say ω_1. This grammar can clearly be used for pattern classification since a given pattern of unknown origin can be classified as belonging to ω_1 if it is a sentence of L(G). Otherwise the pattern is assigned to ω_2. For example, the context-free grammar $G = (N,\Sigma,P,S)$ with $N = \{S\}$, $\Sigma = \{a,b\}$, and production set $P = \{S \rightarrow aaSb,\ S \rightarrow aab\}$, is capable of generating only sentences which contain twice as many a's as b's. If we formulate a hypothetical two-class pattern recognition problem in which the patterns of class ω_1 are string of forms aab, aaaabb, and so forth, while the patterns of ω_2 contain equal numbers of a's and b's (i.e., ab, aabb, etc.), it is clear that classification of a given pattern string can be accomplished simply by determining whether the given string can be generated by the grammar G discussed above. If it can, the pattern belongs to ω_1. If it cannot, it is automatically assigned to ω_2.

The above classification scheme assigns a pattern into class ω_2 strictly by default. If a pattern is found to be an incorrect sentence over G, it is assumed that the pattern must belong to ω_2. However, it is entirely possible that the pattern does not belong to ω_2 either. It may represent a noisy or distorted string which is best rejected. In order to provide a rejection capability it is necessary to determine two grammars, G_1 and G_2, which generate languages $L(G_1)$ and $L(G_2)$. A pattern is assigned to the class over whose language it represents a gramatically correct sentence. If the pattern is found to belong to both classes it may be arbitrarily assigned to either class. If it is not a sentence of either $L(G_1)$ or $L(G_2)$, the pattern is rejected.

In the M-class case we consider M grammars and their associated languages $L(G_i)$, $i = 1,2,\ldots,M$. An unknown pattern is classified into class ω_i if and only if it is a sentence of $L(G_i)$. If the pattern belongs to more than one language, or if it does not belong to any of the languages, it is rejected.

Based on these concepts, it is noted that syntactic recognition is based on assigning a terminal sentence (pattern) to a language. This is accomplished by the use of parsing techniques or by implementation of automata which are derived directly from the grammars in question [58]. Pattern variabilities caused by noise or other corrupting influences are handled by the use of error correction techniques or stochastic grammars [58, 80].

Thus far, only pattern descriptions based on string representations have been discussed. Generalizations of strings, trees, graphs, webs, and plexes [58, 75, 76, 77]. The grammars for handling these structures are basically of the form discussed above, but have a more complex specification in the manner in which the productions are applied. The motivation for using these "higher-dimensional" representations is that they offer a more natural and powerful approach for handling the types of structures found in scenes. It is important to note, however, that, with the exception of trees, the productions and recognition techniques for these pattern representations are quite difficult to formulate and implement.

In the decision theoretic approach to pattern recognition, learning techniques have enjoyed a significant level of development. In syntactic recognition, however, this is still one of the major open problem areas. Learning (called grammatical inference) in this case refers to obtaining a grammar from each class by using sample pattern sentences. Obtaining a grammar that generates exactly the patterns in the training set is a trivial task [58]. The problem arises in trying to generalize the structure of the class from a finite set of samples [81]. The most successful algorithms in this area are based on regular string grammars [78] and expansive tree grammars [79].

2.5.3 Interpretation

In this chapter we view interpretation as a process which assigns meaning to a scene based on the results of recognition. The fact that we have separated these two functions implies that recognition is carried out without the use of scene contextual information. In other words, recognition yields a set of labeled, isolated objects while interpretation uses these objects to produce a description of scene content.

There is no question that this subdivision introduces limitations in the capabilities of a scene analysis system. As indicated earlier, however, integration of scene analysis functions in an interactive, "intelligent" framework is well beyond the state of the art in the field of machine intelligence [82]. Due to this limitiation, the types of interpretation techniques in use today are heuristic and at a relatively low level of sophistication. There are numerous specialized applications, however, where even a limited degree of interpretation based on recognition of important features can have a significant impact. Examples include the categorization of aerial scenes as being either of military or nonmilitary interest, the rejection of faulty electronic components in an assembly line, adaptive locomotion of robots by visual feedback, and autonomous target detection systems.

2.6 Organization of Scene Data

An area of considerable recent interest in scene analysis is the search for techniques suitable for representing and organizing in a systematic way the large amounts of data required for the digital representation of a scene. Innovative development of algorithms for segmentation, recognition, and interpretation involves identifying and exploiting structural relationships prevalent within the sensed data. These requirements lead to the formulation of data representation techniques based on a discrete mathematical framework.

In this section, attention is focused on properties of data structures and databases as they apply to problems in scene analysis. There are at least three strong arguments in favor of using standard data structures (such as link lists, stacks, and trees) and standard database organizations (such as relational, hierarchical, and network models) for representing and organizing scene data. First, these standard data representations have a firm mathematical foundation which can be used as a starting point for new algorithm development. Second, they facilitate the evaluation of important algorithm characteristics such as time complexity and computational requirements. Finally, consistent data representation techniques allow systematic approaches to scene data documentation, storage, access, and manipulation.

2.6.1 Data Structures

This section gives some of the important definitions and concepts in data structures (cf., Knuth [101], Horowitz and Sahni [93]) and provides several examples of the use of such structures as lists, trees, and graphs in digital scene representation and processing.

2.6.1.1 Definitions and Basic Concepts

The cartesian product of set A with set B is denoted by A x B and is the collection of ordered pairs (a,b), a in A and b in B. A subset of A x B, say $R \subseteq A \times B$, is a binary relation from A to B.

If A = B x C, then

$$A \times D = \{((bc),d) \mid b \text{ in } B,\ c \text{ in } C,\ d \text{ in } D\}.$$

Usually, we can view this product as an associative formation of ordered triples and write simply

$$B \times C \times D = \{(bcd) \mid b \text{ in } B,\ c \text{ in } C,\ d \text{ in } D\};$$

then $R \subseteq B \times C \times D$ is a ternary relation. In general, $R \subseteq A_1 \times A_2 \times \ldots \times A_n$ is n-ary relation, $n \geq 1$.

If A = B, then $R \subseteq A \times B = A \times A$ is a relation on the set A. For instance, given $A = \{a_1,a_2,a_3\}$ and $A \times A = \{(a_1,a_1),(a_1,a_2),(a_1,a_3),(a_2,a_1),(a_2,a_2),(a_2,a_3),(a_3,a_1),(a_3,a_2),(a_3,a_3)\}$, then $R = \{(a_1,a_1),(a_1,a_2),(a_1,a_3),(a_2,a_2),(a_2,a_3),(a_3,a_3)\}$ is a relation on A. A relation R on set A is:

(i) reflexive if (a,a) is in R for each a in A;

(ii) symmetric if (a,b) in R implies (b,a) in R;

(iii) antisymmetric if (a,b) and (b,a) in R implies a = b;

(iv) transitive if (a,b) in R and (b,c) in R implies (a,c) in R.

The specific relation R given above on $A = \{a_1,a_2,a_3\}$ is therefore reflexive,antisymmetric, transitive and not symmetric.

Lists

A fundamental data structure is one for which the elements of a finite set A are ordered in a list from the first element to the last. Specifically, a linear list is a set of elements $A = \{a_1,a_2,\ldots,a_n\}$, $n > 0$, for which there is a relation $R \subseteq A \times A$ that is reflexive, antisymmetric, transitive, and has the additional property that, for any two elements a_i and a_j in A, either (a_i,a_j) is in R or (a_j,a_i) is in R. A relation with these characteristics is called a linear ordering. If (a_i,a_j) is in a linear order R, we usually write $a_i \leq a_j$. The first element in the linear list is that element a_i such that $a_i \leq a_j$ for each a_j in A. The last element is that element a_j for which $a_i \leq a_j$ for each a_i in A. When subscripts are used for a linear list of n elements, the standard notation has elements ordered as $a_1 \leq a_2 \leq \ldots \leq a_{n-1} \leq a_n$. The specific relation R on $A = \{a_1,a_2,a_3\}$ given above defines a linear list with the order $a_1 \leq a_2 \leq a_3$.

Often a linear list used as a data structure must be altered during the execution of an algorithm. Special kinds of linear lists are defined by the ways in which operations on them are allowed to be performed. A list is a <u>stack</u> if all insertions and deletions of elements are made on one predetermined end. A list is a <u>queue</u> if all insertions are made on one end and all deletions are made on the other.

In actual computer storage, a linear list of n elements that is static and fixed in size can have the data value(s) of its elements stored in sequential memory locations according to the natural sequencing of elements from first to last. Other techniques are more appropriate for a nonstatic list that must be changed in size or rearranged at certain times. A <u>linked list</u> is created by associating with an element both its data value(s) and a <u>pointer</u> or <u>link</u> to the first location of the next element in the linear ordering. Insertions and deletions of elements can then be handled by changing the necessary link values, as shown in Fig. 2.34.

A <u>circular-linked list</u> is created by linking the last element to the first in a linear list; this allows all elements to be accessed by tracking through the list, independently of which node is used as the initial entry point. The concept of first and last elements is not always enforced for a circular-linked list, but a unique element is sometimes designated the "head" or starting node for entry into the list, as shown in Fig. 2.35.

A <u>double-linked list</u> is one in which there is stored for each element (i) its data value(s), (ii) a pointer to its immediate predecessor in the linear ordering, and (iii) a pointer to its immediate successor in the

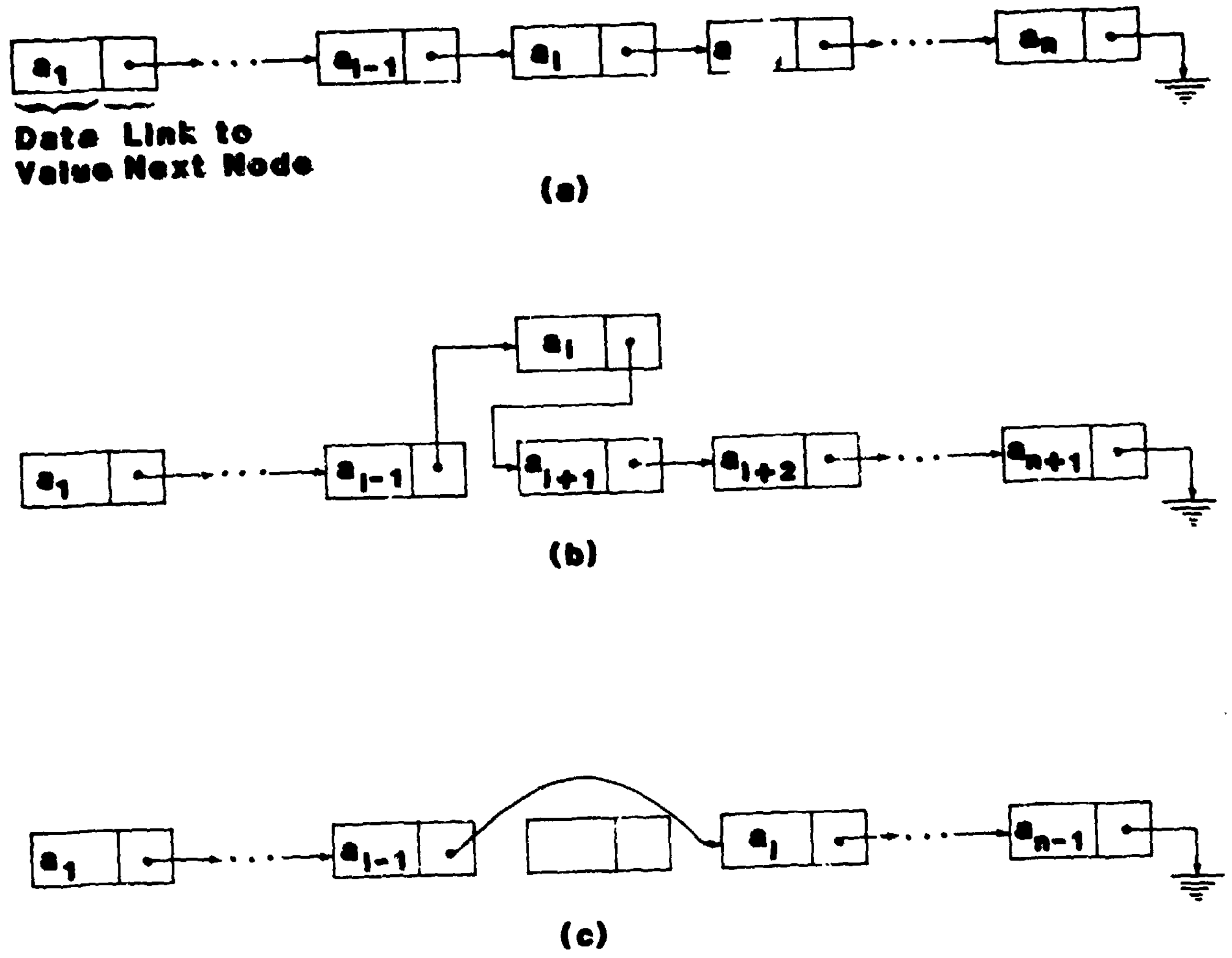


Figure 2.34. (a) Linked list. (b) Node insertion to expand list. (c) Node deletion to shorten list.

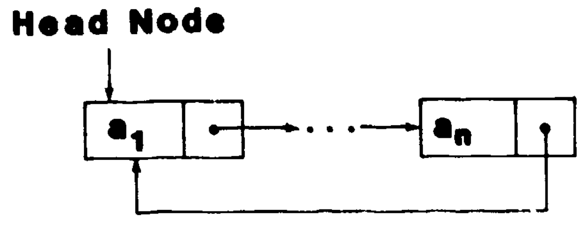


Figure 2.35. Circular-linked list.

linear ordering. A search or trace in this kind of list structure can move easily in either direction for any entry node.

If a linear list is regarded as a set of items each of which has a single subscript, then a generalization is an array of items with multiple subscripts such that any subscript, when taken through all its values in order from first to last while other subscript values are fixed, creates a linear list. For an array, any one of the simple list storage or linking techniques may be used for the individual subscripts, for example, linked lists as shown in Fig. 2.36.

It is also possible to order a static multidimensional array in a one-dimensional, linear way. One method is to impose a standard sequence for stepping through subscript values, as in the FORTRAN convention. A second method employs a relation between the set of subscript values and the integers For example, for a two dimensional array, say A(x,y), with positive integer subscripts, let the relation $R \subseteq (I \times I) \times I$, where I is the set of positive integers, be defined so that ((x,y),z) in R means

$$z = \frac{(x + y - 1)(x + y - 2)}{2} + y$$

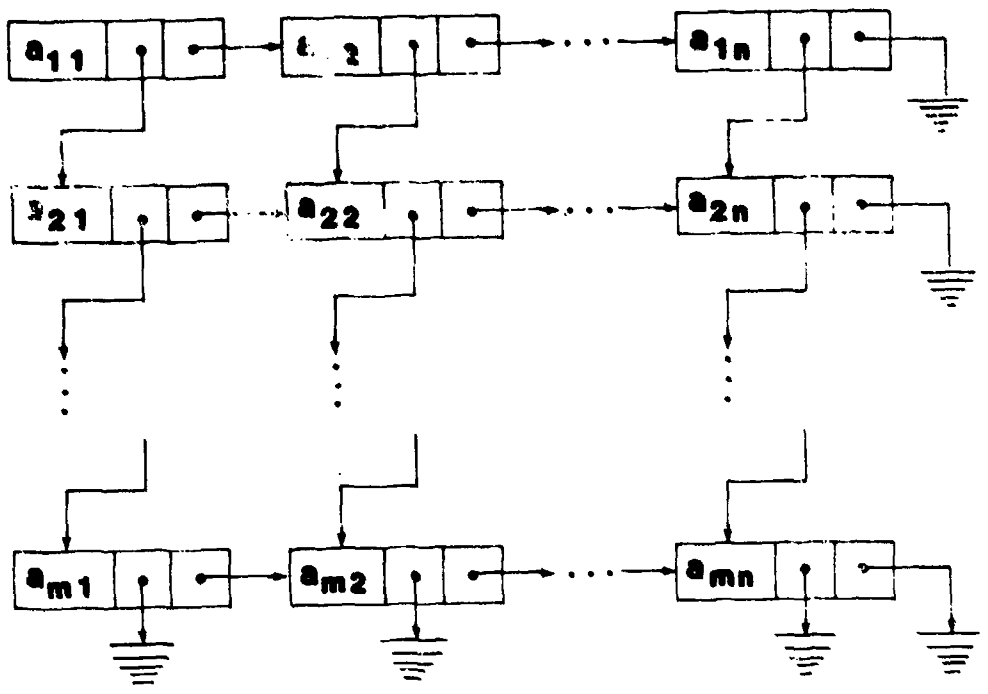
It can be shown that, for a given ordered pair (x,y), z is a unique positive integer (see Fig. 2.37) which can be interpreted as the position of the element A(x,y) in a linear list. This equation can be used recursively for arrays of dimension greater than two.

Trees

A structure more complex than a list is obtained by permitting a node to link to more than one immediate successor. This hierarchical or

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots\vdots\vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

(a)



(b)

Figure 2.36. (a) Two-dimensional array.
(b) Possible array linkages.

		y			
		1	2	3	...
x	1	1	3	6	...
	2	2	5	9	...
	3	4	8	13	...
	⋮	⋮	⋮	⋮	

Figure 2.37. Values of $z = (x+y-1)(x+y-2)/2 + y$.

tree structure occurs frequently in computing. A tree T consists of a set of nodes with the properties:

(i) there is a unique node called the root of T; and

(ii) the remaining nodes are partitioned into m disjoint subsets $T_1, \ldots, T_m$, each of which is a tree called a subtree of T.

It is noted that this is a recursive definition in the sense that it cites itself. The convention is to draw a tree from its root downward to its leaves (a leaf being a node with no offspring or successors) as illustrated in Fig. 2.38. Among the characteristics of a tree T is that there is exactly one directed path from the root to any other node in T, so that each complete path from the root to a leaf follows a linear ordering of a subset of the nodes in which the root is the first element and the leaf is the last.

In order physically to represent a tree, we note that two items are required for each element:

(i) information about a node, stored as a set of values for the node; and

(ii) information relating a node to its predecessor and/or offspring, stored as a set of pointers to these nodes.

Thus, the pointer techniques described above for lists may also be used for trees.

Although a tree is inherently a two-dimensional structure, it is often necessary to represent a tree as a one-dimensional list of all its nodes obtained in a manner consistent with the way in which nodes of other trees would be listed. The fundamental hierarchy of the tree can be used to obtain either a left-bracketed or a right-bracketed representation as

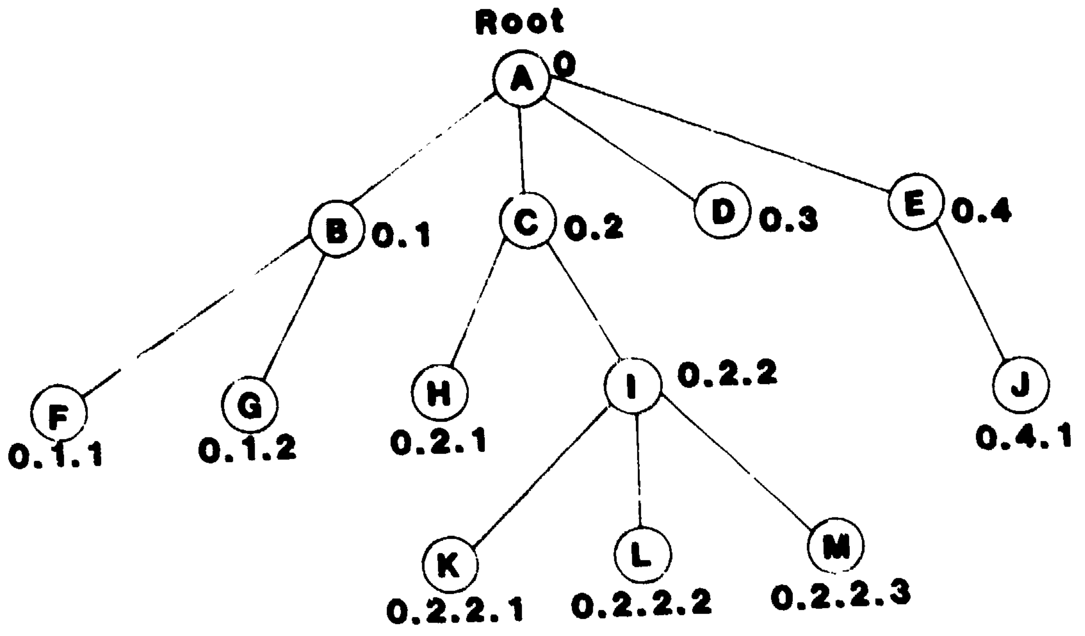


Figure 2.38. Tree T.

follows (Aho and Ullman [83]). For the left-bracketed representation, denoted by lrep(T):

(i) If the root is labelled X and has subtrees $T_1, \ldots, T_m$ in left to right order, then

$$\text{lrep}(T) = X(\text{lrep}(T_1), \text{lrep}(T_2), \ldots, \text{lrep}(T_m)).$$

(ii) If T consists of root X only, then lrep(T) = X.

For the right-bracketed representation, rrep(T):

(i) If the root X has subtrees $T_1, \ldots, T_m$ in order left to right, then

$$\text{rrep}(T) = (\text{rrep}(T_1), \ldots, \text{rrep}(T_m))X.$$

(ii) If T consists of root X only, then rrep(T) = X.

For tree T in Fig. 2.38, lrep(T) = A(B(F,G),C(H,I(K,L,M)),D,E(J)) and rrep(T) = ((F,G)B,(H,(K,L,M)I)C,D,(J)E)A. The linear list of nodes that remain when the brackets are deleted from lrep(T) is the preorder of the elements in T; the linear list obtained by deleting the brackets from rrep(T) is the postorder of the elements in T.

Another representation of the tree hierarchy is obtained by using Gorn addressing for the nodes (Gorn [92]). The indices of T are defined recursively by assigning the index 0 to the root, and if a node with index k has n offspring, they are assigned indices, k,1,k,2, ..., k,n in order from left to right. Given a tree with these indices, there is a lexicographical ordering relation $\underset{L}{<}$ defined on T as follows. Let r and s be indices; then r $\underset{L}{<}$ s if

(i) there is an index c such that s = r.c, or

(ii) r = c.i.u and s = c.j.v where c, u, and v are indices, and i and j are positive integers with i being less than j.

Condition (i) establishes a top-to-bottom ordering along the descendant paths from the root; condition (ii) establishes a left to right ordering among the descendants of any one node.

Figure 2.38 includes the Gorn addresses for the nodes in the tree T. The node labelled K with index s = 0.2.2.1 is a descendant of node I with index r = 0.2.2 because s = r.c with r = 0.2.2 and c = 1. The node labeled H with index r = 0.2.1 is to the left node I with index s = 0.2.2 because r = c.i and s = c.j with i < j.

Graphs

The last discrete mathematical structure we define is a directed graph or digraph G, which is a set A of nodes and a relation $R \subseteq A \times A$. If (a,b) is in R, then there is an edge or arc directed from node a to node b. An acyclic digraph is one that contains no cycles, i.e., no path along directed edges from a node back to itself; thus, a tree is a special case of an acyclic digraph.

An ordered digraph G is defined by a pair (A,R) where A is the set of nodes and R is a set of linear lists of arcs, with each list of the general form $((a,b_1),(a,b_2), \ldots, (a,b_n))$ for $a,b_1, \ldots, b_n$ in A. This list, in its digraph interpretation, means that there are n arcs leaving node a, the first going to node b_1, the second going to b_2, and so on through the last going to node b_n. The tree T in Fig. 2.38, viewed as an ordered digraph, would be defined by the set of nodes {A,B,C,D,E,F,G,H, I,J,K,L,M} and the following lists: ((A,B),(A,C),(A,D),(A,E));((B,F),(B,G)); ((C,H),(C,I)):((E,J)),((I,K),(I,L),(I,M)).

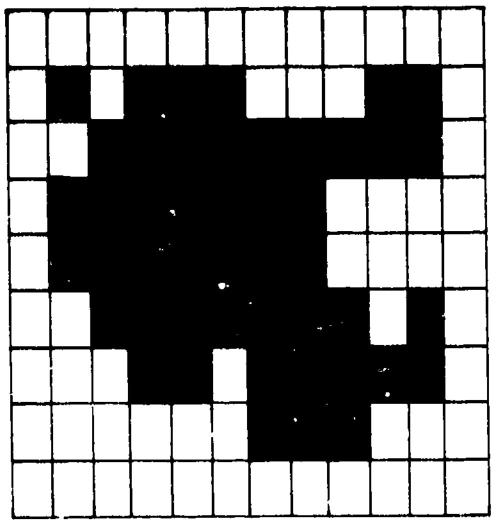
2.6.1.2 Use of Lists for Scene Representation and Processing

Important tasks in many scene processing systems are the detection and representation of boundaries for the regions in an image. A region is generally defined to be a subimage of pixels that are connected by transitivity and share a common characteristic or attribute, such as having the same value of gray level. A pixel that has at least one immediate neighbor which belongs to a different region in any of the eight adjacency directions is called a boundary pixel; otherwise, a pixel is interior to a region. Various algorithms to detect the boundary pixels and to order them as sequences that trace the outlines of the regions may be found in the literature [3, 31, 36, 47]. Circular-linked lists are often used to store the sequenced boundary pixels to allow an efficient trace around the outline of any region in a two-dimensional image.

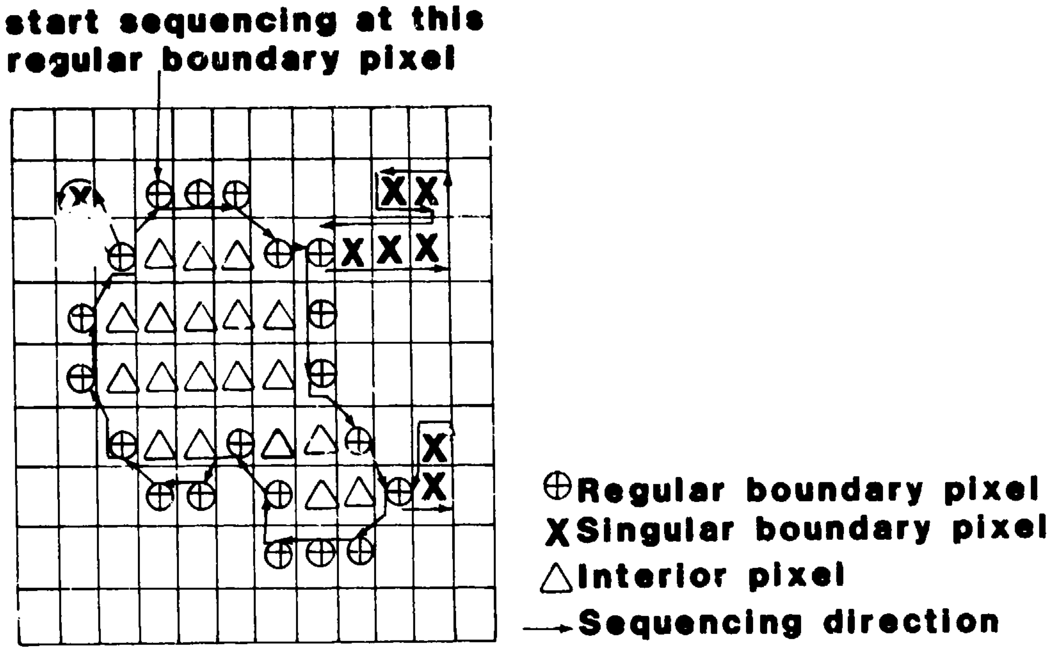
In this section, we illustrate the use of lists for boundary representation and give an example of list processing to compress the boundary representation. An approach due to Lee [103] uses two words of storage for each node in the sequenced boundary list. The first word stores the location of the boundary pixel as a single value z obtained from its (x,y) coordinates by the equation

$$z = x + (y-1) * n$$

where n is the dimension of the n x n image array. The second word is the link that points to the first word of the next node in the boundary sequence. The location value z of a singular boundary pixel (that is, a boundary pixel not adjacent to any of its region's interior pixels, as illustrated in Fig. 2.39) is made a negative number as a flag. The



(a)



(b)

Figure 2.39. (a) Region. (b) Classification of region pixels.

sequencing priority is shown in Fig. 2.40 in which a lower number indicates a higher priority.

Figures 2.41(a) and (b) show two digitized images, Figs. 2.41(c) and (d) are segmented and averaged versions of the images, and Figs. 2.41(e) and (f) are the region boundaries traced and displayed via the circular-linked list representations.

Two methods for compressing the data in the sequenced circular-linked lists for region boundaries have been studied by Lee [103]. Both are based on the eight direction codes (Freeman [51]) corresponding to eight-connective descriptions with respect to a reference boundary pixel, as shown in Fig. 2.42. The codes are assigned pixel by pixel by moving through the circular-linked list from an arbitrarily selected starting pixel, so every node in the list has a direction code from 0 to 7, except for the starting pixel itself.

The first coding scheme sequentially combines every two successive direction codes into one code by using the coding equation,

$$\text{CODE} = (\text{DIRCD1} + 1) * 8 + \text{DIRCD2}$$

where DIRCD1 is the first of the two successive base direction codes and DIRCD2 is the second. CODE is the value of this part of the shape descriptor. If the total number of base direction codes of the boundary is odd, the last direction code cannot be combined into one code using the above equations, so in this case, CODE is just set equal to this last base direction code. Since the value of CODE generated by the equation is always positive, "-128" is used to represent the endmark of the shape

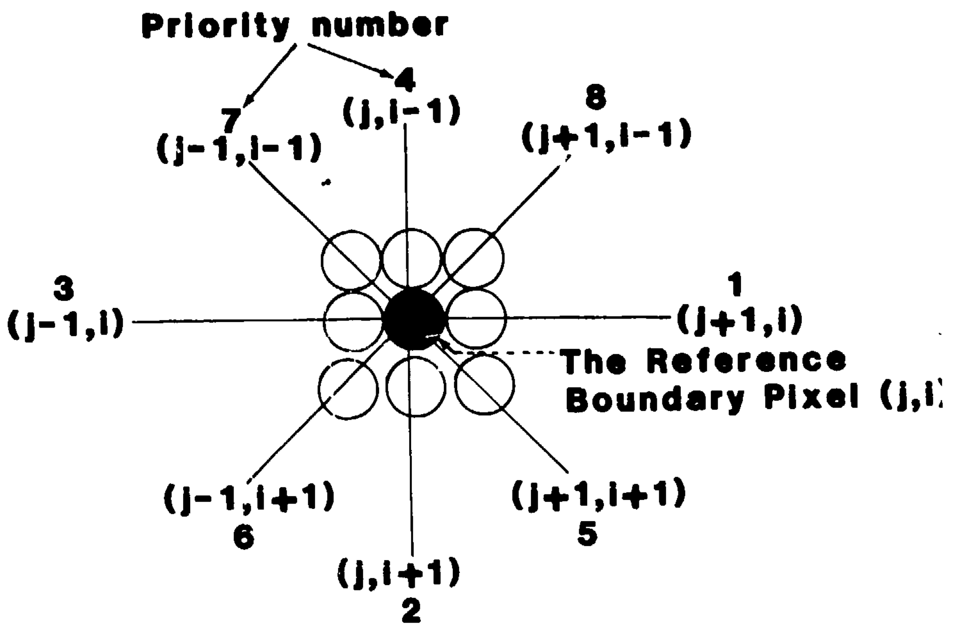


Figure 2.40. Sequencing priorities.

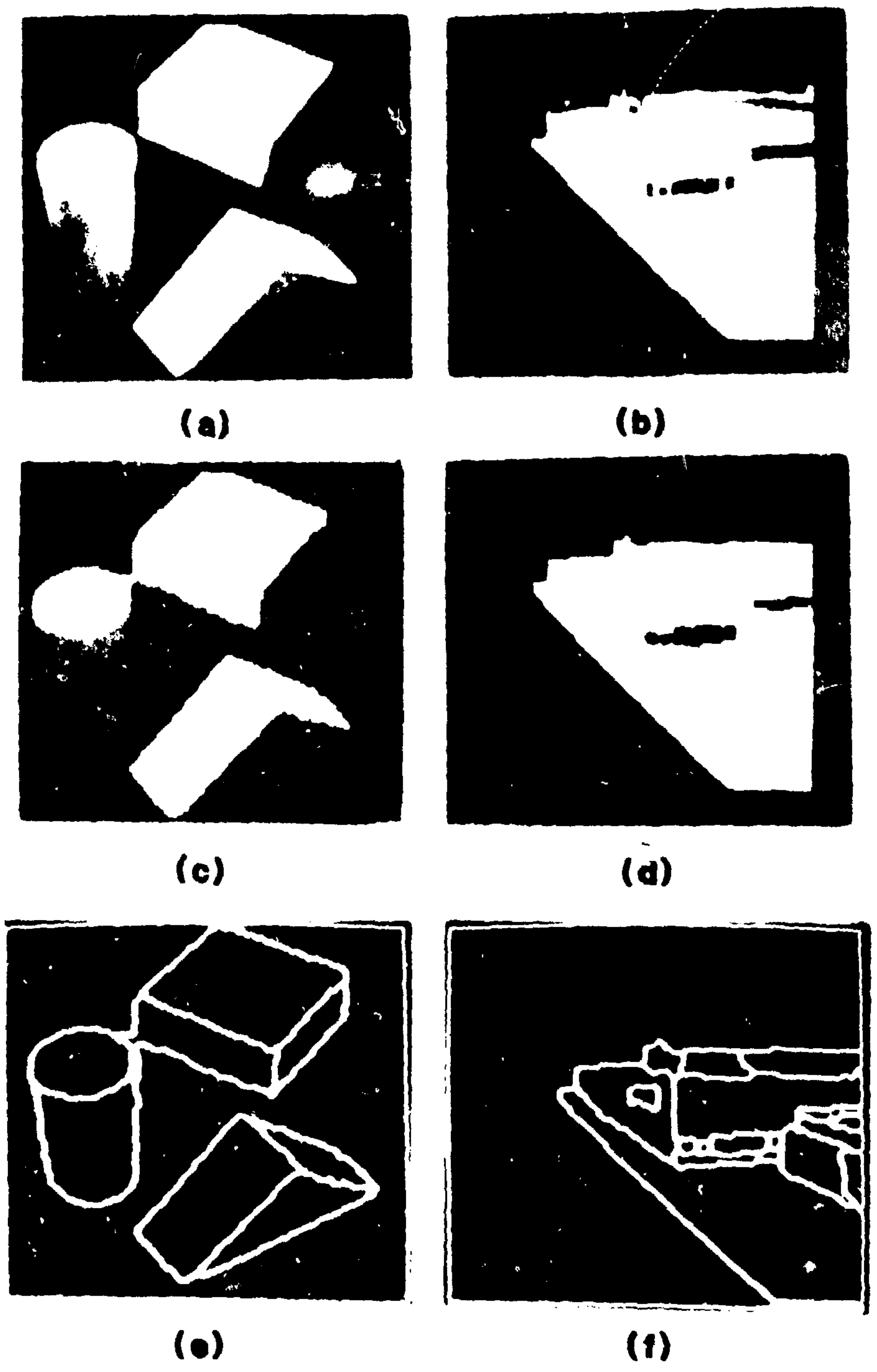


Figure 2.41. (a) and (b) Original images. (c) and (d) Segmented images. (e) and (f) Boundaries from lists of sequenced boundary pixels.

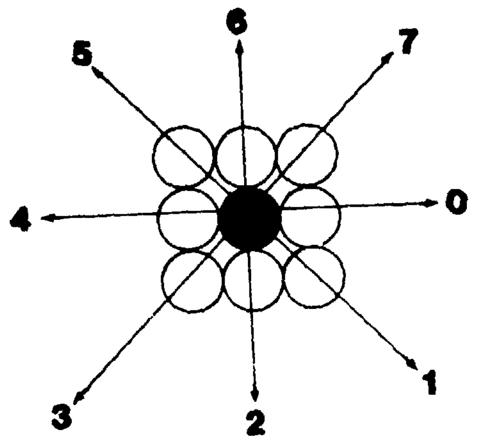


Figure 2.42. Base direction codes.

descriptor. A segment of a sequenced boundary is shown in Fig. 2.43; the shape descriptor is "17,26,8,15,7,-128."

The second coding scheme is basically a run-length coding [3] of the direction vector. A direction vector is a section of a boundary in which all pixels have the same base direction codes. The number of the pixels of a direction vector is called its length. Since the number represented by a byte for the code ranges from -128 to 127, the eight possible kinds of direction vector are divided into two parts in order to code as long a direction vector as possible. Each direction vector is coded by the corresponding coding equations,

$$\text{CODE} = (\text{DIRCOD} * 32) + \text{LENGTH} \quad \text{for } 0 \le \text{DIRCOD} \le 3$$

and

$$\text{CODE} = [(\text{DIRCOD} - 4) * 32] + \text{LENGTH} \quad \text{for } 4 \le \text{DIRCOD} \le 7$$

where DIRCOD represents the base direction code of the direction vector and LENGTH is the length of the direction vector.

The number "0" is used to represent the endmark of the shape descriptor in the second coding scheme. The maximum length of the direction vector is "31" because four different kinds of direction vector with lengths ranging from 1 up to 31 can be represented by the numbers from 1 to 124 (-1 to -124) without exceeding the range from 1 to 127 (-1 to -127). If a direction vector's length exceeds 31, it must be divided into multiple direction vectors. For example, if there is a direction

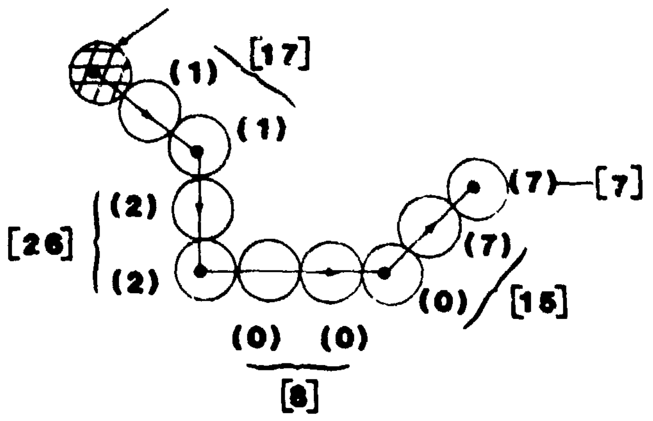


Figure 2.43. Shape description using the first coding scheme. (n) Base direction code. [m] Code in shape descriptor.

vector with length 70, it will be sequentially divided into three direction vectors with lengths 31, 31, and 8. (If the maximum length of the direction vector were allowed to be 32, direction vectors could be coded by the numbers ranging from 0 to 127 and -1 to -128; however, a problem might arise, in that no more numbers would be available to represent the endmarks of the shape descriptors.) A segment of a sequenced boundary is shown in Fig. 2.44; the shape descriptor is "70,5,-98,39,-3,0" using the second coding scheme.

The above compression techniques have been used for processing 128 by 128 images with no more than 256 regions. For the first scheme, two boundary pixels represented by four words in the original list can be represented by one byte after compression, giving a compression ratio approaching 16 to 1. The compression ratio in the second scheme is a function of the lengths of the direction vectors in the image. If the boundaries are relatively "regular" and consist of long runs of pixels with the same direction code, the ratio can approach 248 to 1 because up to 31 boundary pixels can be coded into one byte. In the worse case, the direction vectors have length 1, in which case the ratio is 8 to 1.

2.6.1.3 Use of Quad Trees for Image Representation and Processing

An <u>n-ary</u> tree for $n \geq 1$ is a tree in which each node either has n offspring or is a leaf with no offspring; thus, n = 2 means a binary tree, n = 3 means a ternary tree and n = 4 means a quad tree. Quad trees have been found to be particularly attractive as data structures in certain areas of scene analysis and image processing.

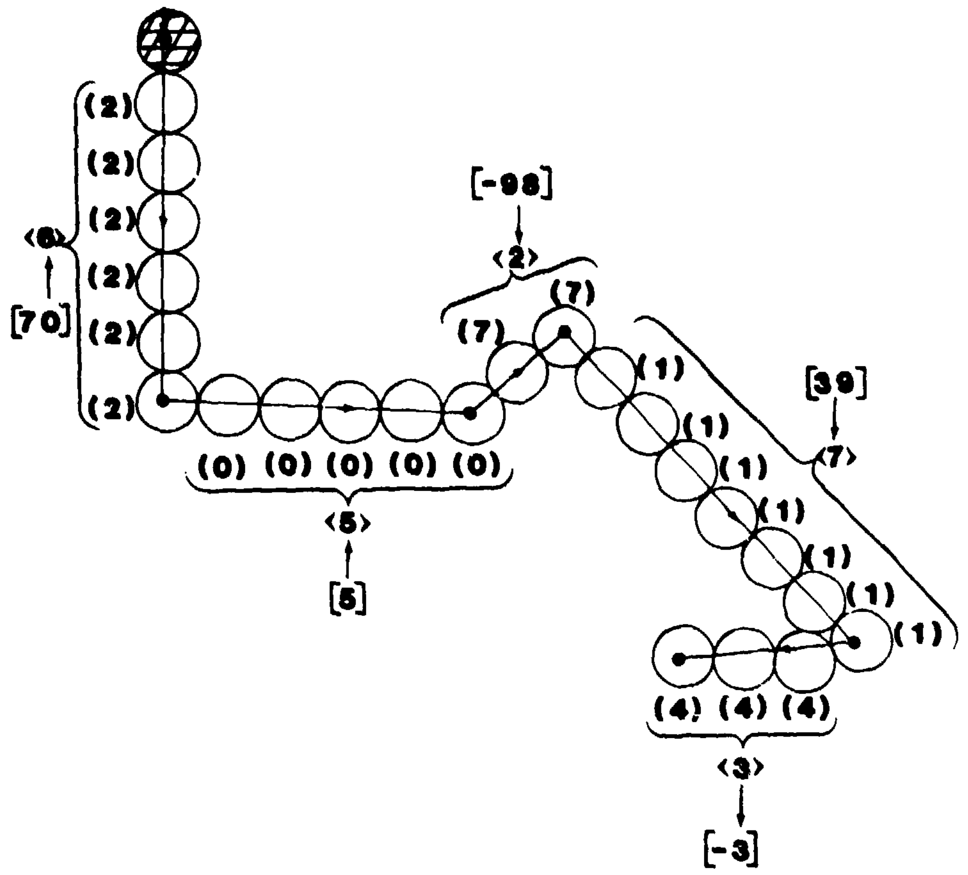


Figure 2.44. Shape description using the second coding scheme. (n) Base direction code. [m] Code in shape descriptor. <k> Length of direction vector.

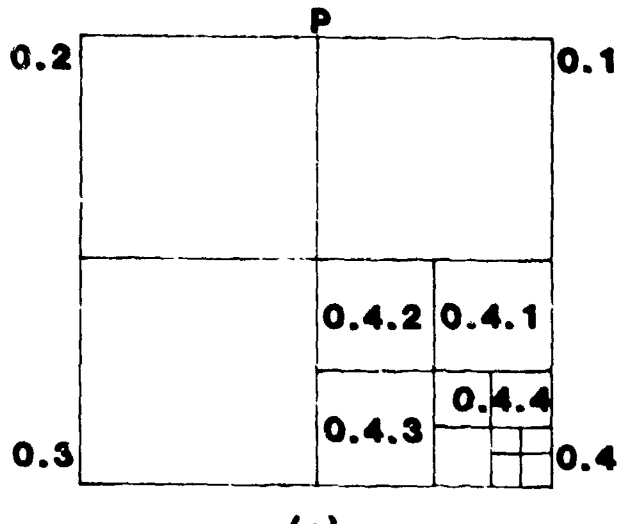
Given an image P as a rectangular array of pixels characterized by quantitative parameters such as gray level or color intensity, we develop a quad tree for P using Gorn addressing in the following way:

(i) The root with index 0 is labeled P.

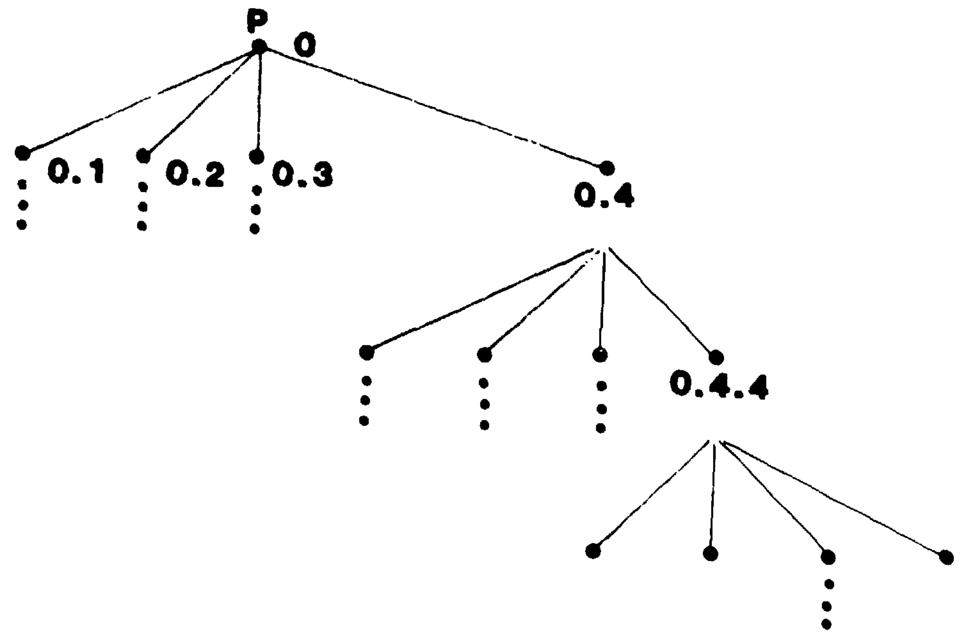
(ii) P is divided into quarters to make subimages corresponding to tree nodes 0.1, 0.2, 0.3, and 0.4. If the parameter(s) for all pixels in the subimage for node 0.i, $1 \leq i \leq 4$, are identical, then node 0.i becomes a leaf labeled with the parameter value(s); if all parameters are not equal, then node 0.i becomes the root of a quad subtree and the subimage 0.i is itself quartered. The test for creating a leaf node is then applied to the subimages 0.i.j, $1 \leq j \leq 4$.

(iii) Step (ii) is carried out until all paths terminate on labeled leaves. Ultimately, of course, a leaf could correspond to an individual pixel.

Figure 2.45 shows the nature of the quad tree hierarchy for an image, and Fig. 2.46 shows the subimages associated with a polygon. A quad tree is an example of a regular decomposition of an image because there is regularity in the shape, size, and parameters of a subimage represented by a given level in the tree. Some of the advantages of a regular decomposition of an image are the creation of subimages convenient for processing if the entire image P is too large for main storage, rapid access to the representation of any geographical part of the image, a hierarchical structure suitable for recursive analysis techniques, and general independence of the structure from the type of image being processed.



(a)



(b)

Figure 2.45. (a) Quartering of image P. (b) Quad tree representation.

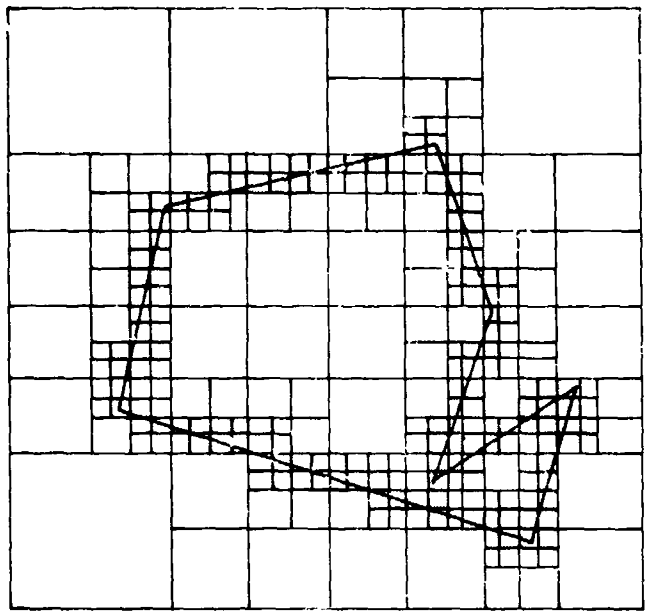


Figure 2.46. Quartering of a polygon.

To carry out experiments in reduction of quad trees, Klinger and Dyer [99] define the relative importance of subimage i.j subimage i as

$$d(i.j,i) = \frac{\text{intensity of } i.j}{\text{intensity of } i} \qquad (2.6\text{-}1)$$

where intensity refers to the average gray level value of the pixels; thus $0 \leq d(i.j,i) \leq 1.0$. Using two thresholds, w_1 and w_2, a subimage i.j is called "informative" if $w_2 \leq d(i.j,i) \leq 1.0$, "questionable" or "not sure" if $w_1 \leq (di.j,i) < w_2$, and "noninformative" if $0 \leq d(i.j,i) < w_1$. In building the quad tree for the offspring of node i, the node i.j is made an empty entry if i.j is noninformative, made a single leaf if i.j is informative, and made a nonleaf node (the root of a quad subtree)if i.j is questionable.

Digitized images representing letters, blocks, and polygons in 64 by 64 and 32 by 32 arrays of pixels with gray levels ranging from 0 to 9 were processed to obtain quad tree structures for the image objects. Illustrations of the kinds of results obtained in the experiment are given in Table 2.3, in which the "% Object Area Lost" is the percentage of pixels that are part of some object which were eliminated from the data structures, and the "% Picture Area Deleted" is the percentage of the picture area deleted, i.e., storage saved, as the data structure was created.

The time and space complexity of a variety of algorithms that perform operations on quad tree representations of images have been investigated by Hunter and Steiglitz [95]. In their model, a picture of image P = (C,A) consists of set C, a finite set of pixel colors, and set A, a square array of pixels. Because of the image-and subimage-quartering

Figure Class	% Object Area Lost	% Picture Area Deleted
Letters	18.0	75.0
Blocks	10.0	64.0
Polygons	4.0	50.0

Table 2.3. Experimental results for thresholds $w_1 = 0.10$ and $w_2 = 0.25$

property of a quad tree, such a tree T corresponds to P decomposed into a 2^q by 2^q array, $q > 0$; and T has at most $q + 1$ levels.

The algorithms are taken to be executed on a random access machine for which a data transfer to or from memory requires a constant amount of time. Storage size is a fixed number of bits adequate to store, for example, the information labeling a node. Time is measured in executions of constant-time operations on pairs of data words, such as multiplication and comparison of numerical values. The complexity of an algorithm with real-valued inputs $x_1, x_2, \ldots, x_n$, is measured as follows: for real-valued functions f and g, if $f(x_1, \ldots, x_n)$ time or space is required, the algorithm is of order $g(x_1, \ldots, x_n)$, denoted by $\mathcal{O}(g(x_1, \ldots, x_n))$, if $f(x_1, \ldots, x_n)$ is not greater than $kg(x_1, \ldots, x_n)$ for some constant k and for all $x_1, \ldots, x_n$ greater than some fixed integer.

As an illustration, a "condensation algorithm" to eliminate any instances in which the four offspring of a node have the same color in a quad tree can be described essentially as a postorder traversal of the nodes. Specifically, we start at the root and recursively traverse the subtree rooted at each offspring. For a root which is itself a leaf, we attach the leaf color to the root; for a root of a nontrivial subtree, if all four offspring have the same color label, we color the root their color and remove them. This algorithm visits each node only once and therefore requires time that is a linear function of the number of nodes in the original tree.

This technique is representative of a large number of efficient tree-processing algorithms that are recursive and perform various operations on a tree (including modifications of its structure) as nodes and

and subtrees are traversed. Often the specific modifications done at a certain node are determined by the information passed from the frontier upwards to the offspring of the node, that is, the information flows from the bottom upwards through the tree's hierarchy.

A second algorithm described by Hunter and Steiglitz [95] creates an image that is the ordered superposition of N trees representing N pictures of the same size. One color is designated to be transparent to the other colors; otherwise, the nontransparent colors of picture x on top of picture y dominate in the resulting image. Let Super(x,y) be the pairwise superposition of upper tree x on lower tree y. The algorithm traverses both trees x and y in parallel and modifies y wherever necessary to create the new image tree. One of three cases determines the actions taken when a leaf is encountered in either tree:

(i) The traversal visits leaves in both x and y. If the x leaf is transparent, the y leaf is not changed; if the x leaf is not transparent, its color replaces the color of the y leaf.

(ii) The traversal visits a leaf in x and an interior node in y. If the x leaf is transparent, nothing is done; otherwise, the y node and its descendants are replaced by the color of the x leaf. The traversal is continued as if the y node's descendants had been explicitly visited.

(iii) The traversal visits an interior node in x and a leaf in y. In this case, the y leaf is replaced by the subtree rooted at the x node, and all transparent leaves in that subtree are given the color of the y leaf.

If the input and output data for the Super(x,y) algorithm must be transferred to and from memory, then the order of Super(x,y) is $\mathcal{O}$(total number of nodes in x and y).

As a generalization to N trees ordered as N,N - 1, ..., 2,1 for superposition, we compute Super(N,Super(N - 1, Super(N - 2, ...,Super(3, Super(2,1))...))) as a series of pairwise superpositions, then apply the "condensation algorithm" described above to the result. Each computation of Super can be done in time and storage; that is, linear in the number of nodes of its upper tree; the total computation can be done in time and space linear in the number of nodes in all trees. Other picture combinations (union, intersection, etc.) can be performed by similar techniques on quad trees.

An interesting and effective use of quad trees is in dealing with images containing polygons, as shown in Fig. 2.46. Basically, a polygon consists of a collection of vertices characterized by pairs of coordinates, with nonintersecting edges as line segments between consecutive vertices. In a quad tree for a picture of a polygon, the nodes are partitioned into three classes: boundary, interior, and exterior.

In order to describe the complexity of quad tree algorithms for polygons, we assume that the picture P = (C,A) is a 2^q by 2^q array of square, colored pixels. q is the base-2 logarithm of the length of a side A. Let v be the number of vertices in the polygon and p be the smallest integer number of pixel widths not less than the polygon's perimeter. The following results are proved by Hunter and Steiglitz [95].

(i) There are no more than $\mathcal{O}(p + q)$ nodes in the quad tree for a polygon. (The actual bound on the number of nodes is $16q - 11 + 16p$).

(ii) Time $\mathcal{O}(v(p + q))$ is sufficient to construct the quad tree for a polygon, beginning with the original picture $P = (C,A)$.

(iii) Space proportional to the size of the input and output is sufficient to construct the quad tree for a polygon.

Among further results on quad trees, it is shown that there is an algorithm for tree construction for polygons that is asymptotically $\mathcal{O}(v)$, and that this is an optimal algorithm in the limit of increasing v when the resolution q is fixed.

A method of image segmentation based on traversal of a quad tree has been proposed by Horowitz and Pavlidis [94]. In a pure "merging" method for segmentation, a large number of small picture regions are merged to form larger regions; in a pure "splitting" method, the picture is recursively divided into smaller regions. This technique attempts a direct two-dimensional segmentation by a combined split-and-merge approach.

Suppose X is the domain of a picture, for example, a square array of pixels. Let $f(x,y)$ be the function assigning values to pixels; here $f(x,y)$ is taken to be the brightness function. For purposes of setting the segmentation criteria, a logic function $P:2^X \to \{\text{TRUE},\text{FALSE}\}$ is defined on the power set of X such that

$$P(S) = \begin{cases} \text{TRUE} & \text{if there is a constant } a \text{ such at } |f(x,y) - a| \leq e \text{ for all points } (x,y) \text{ in } S \\ \text{FALSE} & \text{otherwise} \end{cases} \qquad (2.6\text{-}2)$$

where e is a preset threshold factor. A <u>segmentation</u> of X is a partition into subsets S_i, $1 \leq i \leq m$, where, by the definition of a partition,

$$\text{(a) } X = \bigcup_{i=1}^{m} S_i \qquad \text{(2.6-3)}$$

$$\text{(b) } S_i \cap S_j = \phi, \ i \neq j$$

and, in addition

$$\text{(c) } P(S_i) = \text{TRUE}, \ 1 \leq i \leq m \qquad \text{(2.6-4)}$$

$$\text{(d) } P(S_i \cup S_j) = \text{FALSE}, \ i \neq j$$

provided that S_i and S_j are adjacent in X. Note that there may be more than one segmentation of picture X for a given P. A merging algorithm would begin with regions satisfying (c) and merge to create regions satisfying (d). A splitting algorithm would begin with regions satisfying (d) and split to achieve (c). The method of Horowitz and Pavlidis begins with a partition that does not necessarily satisfy either (c) or (d) and produces a partition satisfying both.

In the quad tree structure used, each node corresponds to a square picture region. From an arbitrary initial tree, such a region may be merged with its four adjacent neighbors if all have agreement within the error threshold of their pixel brightness values, or a region may be split into four quarters if there is greater variation in the brightness values. These split-and-merge steps must be followed by a grouping procedure to eliminate artificial region boundaries that might arise from the original segmentation; in this grouping step, adjacent nodes with different predecessors and at different levels of the tree may be merged together (and the regions they represent thereby merged together), provided the segmentation criterion P remains satisfied.

2.6.2 Databases

This section presents some of the important definitions and concepts in contemporary database design and provides several illustrations of the relevance of databases to various aspects of scene data organization and processing. Because a database is generally a high-level organization of large amounts of data, the choice of a database format often implies the designer's own version (or "world model") of the ways in which the information within a scene itself is organized.

2.6.2.1 Definitions and Basic Concepts

We can informally define a database (DB) to be a large collection of data organized according to some predefined structural model. A database management system (DBMS) is a software-hardware organization that permits users to create, access, and/or modify a database. A data definition (DD) language is a high-level language used to establish a database and to define its entities and the relationships among them. A data manipulation (DM) language or query language is a high-level language used to extract or modify data in an existing database.

The decade of the 1970's has been an era of intense study of databases, both in theory seeking unifying concepts and terminology and in practice with various implementations. Generally speaking, a computer system designer expects one or more of the following characteristics to be obtained by an adequate database system (DBS) design (Date [90]):

(i) Reduction of redundancy in the data.

(ii) Reduction or elimination of inconsistency in the data.

(iii) Standard, well-documented procedures for using data.

(iv) Convenient methods for a variety of users to share the data.

(v) Maintenance of data integrity, i.e., assurance that only accurate data is available.

(vi) Assurance that, in the tradeoffs necessary in any digital system design, the aspects of data storage and manipulation have received direct attention.

(vii) Enforcement of data security requirements.

The operating environment in which a DBS is to be used determines the relative weight of each of the above seven considerations. For database design in scene analysis systems, one would virtually always give items (i), (ii), (iii), (v), and (vi) high weights. For a system dedicated to a specific task. item (iv) might not apply; and for a system located in a secure facility, item (vii) might be achieved automatically.

The field of database systems does not as yet have a complete set of accepted nomenclature; however, three areas for concentrated investigation in DBS design have emerged, namely:

(i) The physical and conceptual descriptions of a DBS.

(ii) The three fundamental DB organizations as relational, hierarchical, and network models.

(iii) The DD and DM languages and other DBMS requirements.

Certainly these areas are not disjoint, nor are they necessarily all inclusive for each specific set of needs; but in so far as a unified DBS analysis/synthesis procedure can be said to exist today, it exists in dealing with questions arising in the above three areas. We use these areas as a framework to consider database design for image and scene processing systems.

Physical and Conceptual Descriptions

Figure 2.47 is a standard block diagram of a general DBS. The physical database refers to the actual storage of information as fields, records, files, pages, and perhaps data structures such as lists or trees; thus, it includes characteristics of the storage devices themselves and the stored items such as, for example, the organization of memory pages. The conceptual database is the user's model of the physical database after a syntax-directed translation has been applied to give an interpretation to the stored items; thus, it could include aspects such as, for instance, "the database is a relational model of an image in which rows are segmented regions for which the third attribute is average gray level." A user might then further interpret the information.

Relational, Hierarchical, and Network Models

Three models of a DBS provide the general organizational schemes currently available.

In the relational DB model, data is organized as a table, the rows of which are the individual DB entries and the columns of which are the attributes of the entries. This model is particularly important in image DB design because it places fundamental emphasis on the concept of relations among image components and lends itself naturally to the representation of those relationships. The format for storage frequently used for a relational DB is as a file of fixed-length records in which each

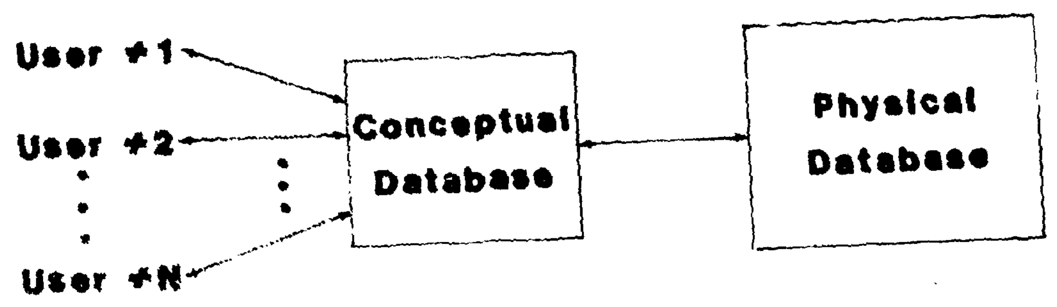


Figure 2.47. General database model.

record is an individual entry and the fields of the record define the attributes.

Figure 2.48 is an illustration of a relational DB table containing information about two "snapshot" images, the first image having four identified objects and the second with three. A generic name appears in the third column as an object attribute. The "superposition order" attribute in the fourth column is a nonnegative integer giving an ordering of objects with respect to the relative distance of an object's center of mass from the image sensor. Thus, the data available about image #1 is that it consists of a CHILD, a TREE TRUNK, and a SHADOW embedded in a BACKGROUND with the CHILD and the TREE TRUNK at the same distance and the SHADOW at a different location relative to the sensor.

For purposes of retrieving data, a <u>key</u> for a relational DB is a subset of the columns with the property that values specified for those columns serve to locate a unique row. For instance, in the table in Fig. 2.48, the set of keys includes [SNAP#, OBJECT#], [SNAP#, OBJECT NAME], and [OBJECT#, OBJECT NAME, SUPERPOSITION ORDER]. A <u>candidate key</u> is a key without redundant attributes (SUPERPOSITION ORDER is redundant in the third key just listed because [OBJECT#, OBJECT NAME] is a key). A candidate key selected for use in accessing data is a <u>primary key</u>; other keys are called <u>nonprimary</u> or <u>secondary</u>.

The organization of a relational DB makes it convenient to form logical combinations of keys for queries. For example, to respond to a user's request for all images in Fig. 2.48 containing a CHILD but not a SHADOW, the DBMS could first find all images without SHADOW as an

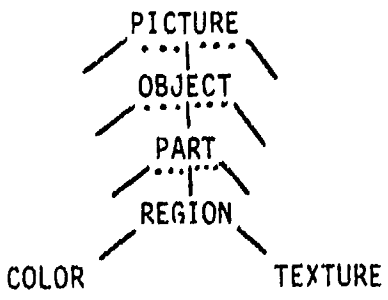
SNAP#	OBJECT#	OBJECT NAME	SUPERPOSITION ORDER
1	1	CHILD	1
1	2	TREE TRUNK	1
1	3	SHADOW	2
1	4	BACKGROUND	0
2	1	LADY	1
2	2	CHILD	1
2	3	BACKGROUND	0

Figure 2.48. Image relational database.

OBJECT NAME value, then search those images for an occurence of CHILD as an OBJECT NAME value.

In the hierarchical DB model, data is organized as a tree in which the root is the least detailed entry and a leaf on the tree frontier contains the most specific type of information. In this model, an entry takes on its full significance only in the context of its superior and its descendant nodes. This model is most effective for data in which a natural tree hierarchy is known or can be approximated. For example a hierarchical DB model is the basis for IBM's general Information Management System (IMS) (Date [90], Ullman [119]). Often a hierarchical DB is stored as a file of variable-length records: one record serves for each DB entry, and its record length is determined by its attributes and its links to other entries. Figure 2.49 is such a model for the tree in Fig. 2.38.

Symbolic or semantic models of images, in which generic names are attached to collections of real image pixels determined to be objects or regions of interest, almost always are considered to have a representational hierarchy determined by the level of the primitive components. Thus, those image processing systems that support research in scene understanding or knowledge acquisition include a hierarchical DB defined explicitly or implicitly. For example, a picture description DB described by Kunii et al.[102] is based on an image or picture hierarchy as follows:



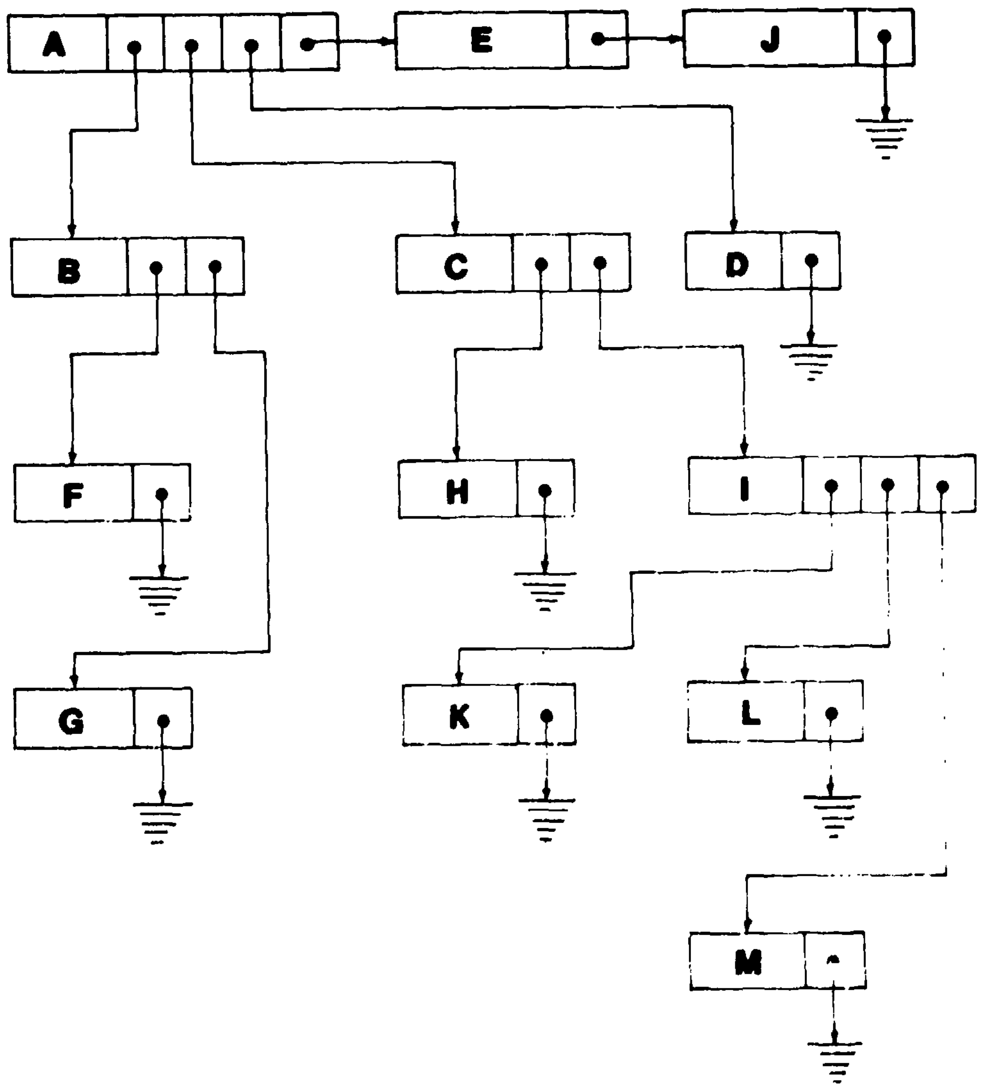
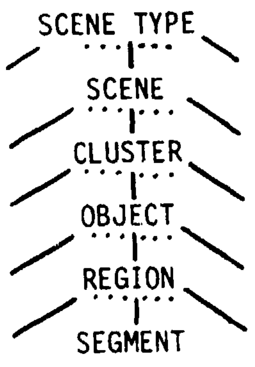


Figure 2.49. Linkages for hierarchy of Fig. 2.38.

There is also a relational DB associated one-to-one with each level in the hierarchy: the picture relational DB is a table of all pictures; for each picture, there is a table of objects such as Fig. 2.48; and so on. Another hierarchy described by McKeoun and Reddy [107] has six levels:



In the network DB model, data is essentially organized as a directed graph, the nodes of which are data records and the arcs of which are record-to-record links. As such, it can be considered a generalization of the tree of the hierarchical model for the case in which there is not a unique hierarchical relationship. This is the organization of some proposed general purpose DBS's (cf., Ullman [119]). A network DB can be stored as a file of variable-length records; and often used is a multilist in which each entry's record has as many pointers to other records as the type of entry has links.

With respect to ease of use by an applications programmer, the relational model has the advantage that it is conceptually the least complex and requires a user to learn the least number of organizational details. With respect to efficiency of implementation (including creation of the DBMS and its expected speed of execution), the two other models perhaps rate higher; in effect, these models often have explicit linkage information that must be inferred in the relational model.

Design of the DBMS

The DBMS includes the systems software necessary to compile and execute data definition/manipulation commands, as well as any hardware designed specifically to support these tasks. In general, the DBMS consists of the operating system interface, DB language compilers and interpreters, and memory management software. In the case of DB organization for scene analysis and image processing, the commands for the definition, construction, modification, and querying of image DB's must be accepted, translated, and executed by the DBMS. The DBMS should assume the burden of the memory operations for data arrays that are too large to reside simultaneously in the main memory.

The advantages of an adequate DBS design include reduction of redundancy in the data and consistency of format. In addition, data independence can be achieved because the DBMS handles the translation of operations expressed in a language convenient to users into detailed database manipulations; in other words, the details of storage, formatting, file manipulation, and other characteristics of the physical database do not have to be known to users who issue processing commands in a high-level language.

2.6.3 Use of Databases in Hierarchical Scene Analysis

An implementation of the basic scene analysis model shown in Fig. 2.3 requires that data be organized in a systematic manner at all levels of the hierarchy. The most commonly used method for doing this is to arrange the labeled information at each level in the form of a relational table. As indicated in the previous section, a typical structure for these tables contains an identification of each element (for example, by number), a list of attributes for each element, and a list of relationships between all elements in the table.* An example is shown in Fig. 2.50 and Table 2.4, where the elements in the relational table are edges, the attributes of individual edges are direction and length, and the relationship between edges is "connected." By examining the first row, for instance, we see that edge number 1 is horizontal, has length L1, and is connected to edges 2, 3, and 4.

As a more complete illustration of these concepts, consider the image shown in Fig. 2.51. In order to simplify the explanation, we have placed a coarse grid on the image such that every vertex of interest coincides with a grid coordinate. In practice, one would use the finer resolution provided by the digitized image itself. With reference to the model of Fig. 2.3, let us assume that the primitives used in the analysis of Fig. 2.51 are edges; then the purpose of the lowest-level recognizer is to extract and label these edges. The results of recognition would be organized in a relational table of the form shown in Table 2.4, in which each row represents a labeled primitive in the hierarchy of Fig. 2.3.

The next higher level in the hierarchy involves segmentation and recognition of parts. In this example, the parts are geometrical surfaces or regions, and recognition is basically reduced to a shape analysis

*Formally, the relations between elements are defined by columns in a table--the way that all attributes are handled in the relational DB model. However, we identify and label these structural relationships separately in our tables to emphasize their role in scene analysis.

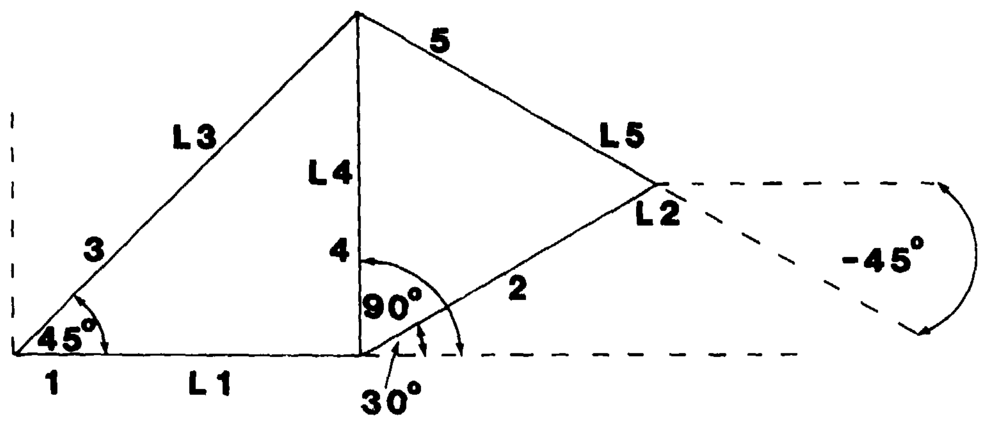


Figure 2.50. Characterization of edges in an object.

Element Number	Attributes		Relationship
	Direction	Length	Connected to
1	0°	L1	2,3,4
2	30°	L2	1,4,5
3	45°	L3	1,4,5
4	90°	L4	1,2,3,5
5	-45°	L5	2,3,4

Table 2.4. Relational table for the edges in Fig. 2.50.

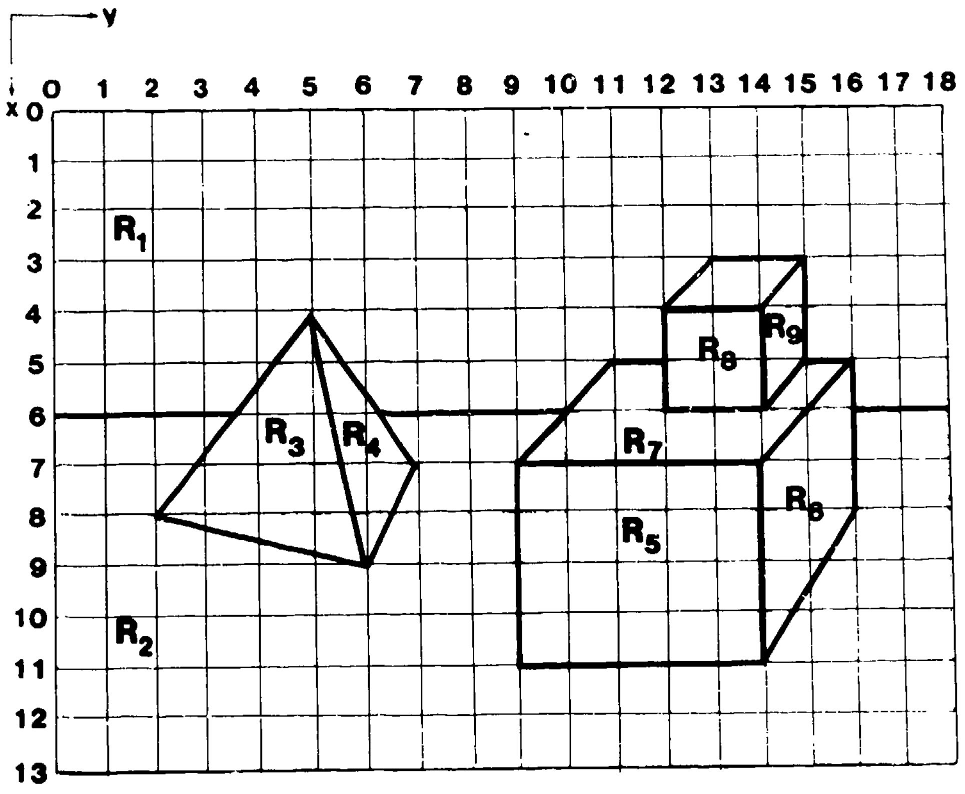


Figure 2.51. Image projection of a scene.

problem. Assuming for the purpose of illustration that region R_1 is red, region R_2 is white, the pyramid is blue, both cubes are red, and all regions (with the exception of R_7) are smooth, the labeled information extracted at this level would be organized in a relational table of the form shown in Table 2.5, witn each row in the table representing a labeled part in the hierarchy of Fig. 2.3. It is noted that some of the information (such as the location of edges) computed by the previous stage of recognition would be used in the assignment of attributes to each of the scene parts.

The extraction and labeling of objects would use the labeled primitives and parts, as well as other sensor information such as range data, the latter being useful for determining the relative three-dimensional location of objects. In this example, the object recognition problem is one of analyzing the shape, properties, and relationships of geometrical volumes. The results would be summarized in a table of the form shown in Table 2.6. As before, each row in the table represents a labeled object in the hierarchy of Fig. 2.3.

The next level in the hierarchy involves grouping objects according to some clustering criterion. For the purpose of illustration, suppose that objects are grouped only if they have the same color and geometrical characteristics. These criteria lead to only one group containing the small and medium red blocks. The relational table for this level of the hierarchy is shown in Table 2.7. In this case, information required to group the objects was already available from the previous level in the hierarchy.

Element No.	Attributes				Relationship
	Shape	Color	Texture	Boundary	Connected to
1	Rectangle	Black	Smooth	(0,0)-(0,18)-(6,18)-(6,0)	2
2	Rectangle	White	Smooth	(6,0)-(6,18)-(13,18)-(13,0)	1,3,4,5,6
3	Triangle	Blue	Smooth	(4,5)-(9,6)-(8,2)	2,4
4	Triangle	Blue	Smooth	(4,5)-(7,7)-(9,6)	2,3
5	Rectangle	Red	Smooth	(7,9)-(7,14)-(11,14)-(11,9)	2,6,7
6	Rectangle	Red	Smooth	(5,16)-(8,16)-(11,14)-(7,14)	2,5,7
7	Rectangle	Red	Rough	(5,11)-(5,16)-(7,14)-(7,9)	5,6,8,9
8	Rectangle	Red	Smooth	(4,12)-(4,14)-(6,14)-(6,12)	7,9,10
9	Rectangle	Red	Smooth	(3,15)-(5,15)-(6,14)-(4,14)	7,8,10
10	Rectangle	Red	Smooth	(3,13)-(3,15)-(4,14)-(4-12)	8,9

Table 2.5. Relational table of parts for the scene shown in Fig. 2.51.

Element No.	Attributes				Relationships		
	Type	Color	Texture	Size	In front of	Right of	Resting on
1	Rectangle	Black	Smooth	Large			
2	Rectangle	White	Smooth	Large	1		
3	Pyramid	Blue	Smooth	Medium	1		2
4	Cube	Red	Smooth	Medium	1	3	2
5	Cube	Red	Smooth/Rough	Small	1	3	4

Table 2.6. Relational table of objects for the scene shown in Fig. 2.51.

Element No.	Attributes				Relationships		
	Type	Color	Texture	Size	In front of	Right of	Resting on
1	Rectangle	Black	Smooth	Large			
2	Rectangle	White	Smooth	Large	1		
3	Pyramid	Blue	Smooth	Medium	1		2
4	Blocks	Red	Smooth/Rough	Medium	1	3	2

Table 2.7. Relational table of object clusters for the scene shown in Fig. 2.51.

As shown in Fig. 2.3, the highest level in the hierarchy is the scene itself, which is composed of labeled object clusters. This level has only one labeled node, the root of the hierarchical tree. Typically, attributes associated with the root node would include scene type and sensor(s) from which the hierarchical representation was derived. In the present example, the scene is a "block-world" scene and the sensors used in obtaining the information for the hierarchical representation were optical and range sensors.

2.6.4 Use of Relational Tables for Three-Dimensional Object Location

An important aspect of scene analysis as applied to computer vision is the location of objects and the interpretation of three-dimensional relationships between these objects. In this section, we illustrate how relational tables may be applied to this problem.

A widely used approach for locating an object in three-dimensional space is the triangulation method (Duda and Hart [59], Yakimovsky and Cunningham [121]). This technique uses two or more image views of the object and, by employing geometric relations and a model of the viewing system (e.g., a TV camera), infers the location in three-dimensional space of corresponding image points. The basic problem is illustrated in Fig. 2.52. Points O_1 and O_2 are the focal points of two cameras which produce images I_1 and I_2 of the three-dimensional body shown in the figure. Point P_1 in this body is projected as point Q_1 onto image I_1 and as point F_1 onto image I_2. Similarly, point Q_2 and F_2 are the projections of P_2 onto these images. It is noted that any point contained in the ray $O_1P_1S_1$ will be imaged as point Q_1 on I_1. However, by

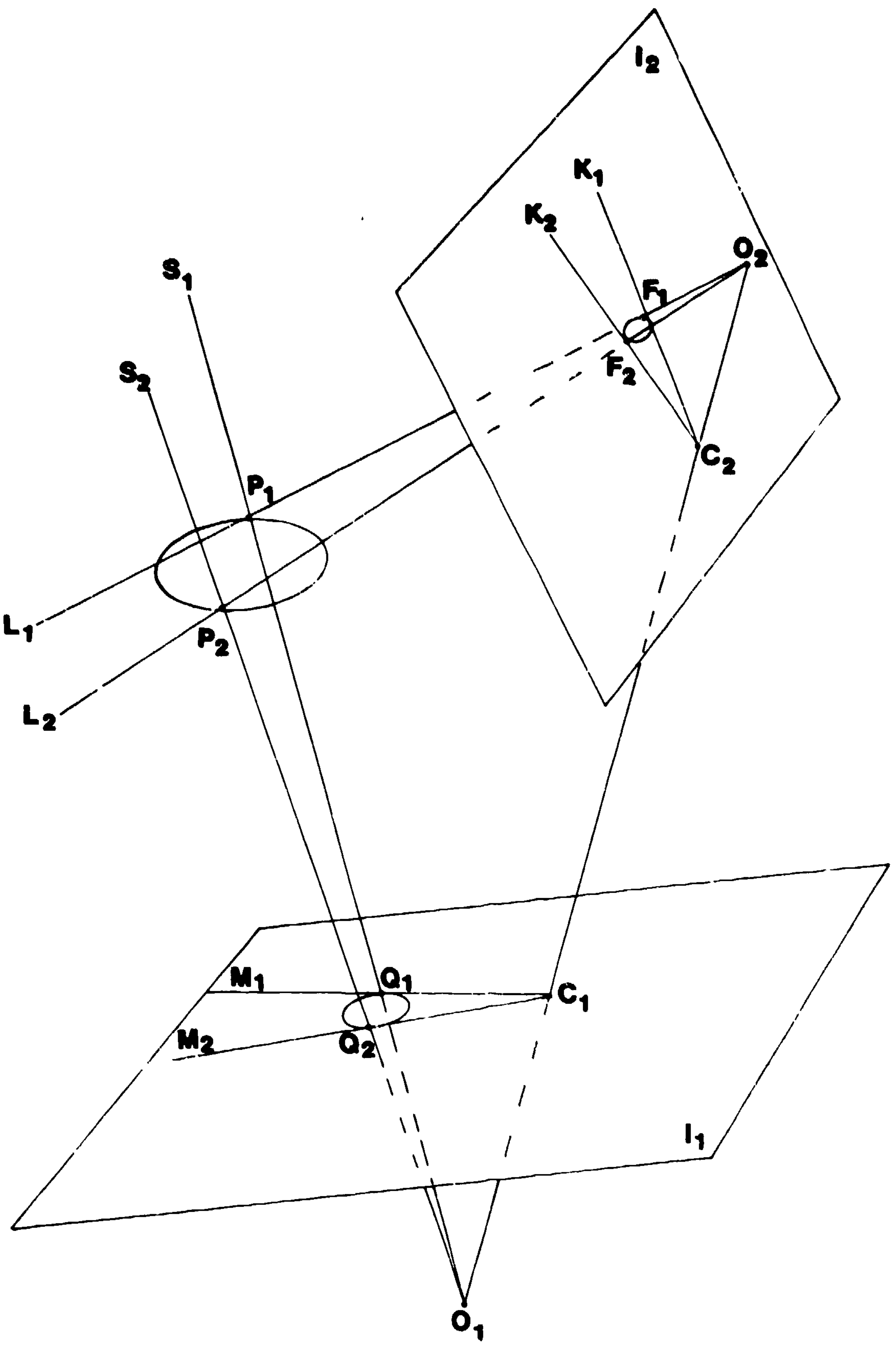


Figure 2.52. Two views of a three-dimensional object.

considering triangle $O_1O_2P_1$, we see that this ray will be imaged as line $C_2F_1K_1$ on I_2. Thus, the point I_2 which corresponds to Q_1 in I_1 will lie on the line $C_2F_1K_1$. Similarly, the point I_1 which corresponds to point F_1 in I_2 will lie on the line $C_1Q_1M_1$. It can be shown (Hwang [96]) that a point I_1 with image coordinates (x_1,y_1), and its corresponding point in I_2 with coordinates (x_2,y_2), satisfy the equation

$$\begin{aligned} f(x_1,y_1,x_2,y_2,k_1, \ldots, k_9) &= (k_1x_1 + k_2y_2 + k_3)x_2 \\ &+ (k_4x_1 + k_5y_1 + k_6)y_2 + (k_7x_1 + k_8y_1 + k_9) = 0 \end{aligned} \qquad (2.6\text{-}5)$$

where the coefficients k_i, $i = 1,2, \ldots, 9$, are determined from geometrical considerations. Hwang has investigated the use of relational tables in conjunction with the above equation for determining corresponding points in two images of the same body. Once these corresponding points are known, their three-dimensional coordinates are easily obtained by standard perspective transformations (Newman and Sproull [108]). These results can then be used to locate and describe the object in three-dimensional space. The advantage of this approach is that it uses the structural information in two views to eliminate the need for a camera model in the three-dimensional reconstruction process.

Hwang's approach is based on extracting regions in images I_1 and I_2 and describing the boundaries of these regions by vertices and their interconnecting edges. The general form of the relational table used to tabulate this information is shown in Table 2.8. The portion of the table above the dashed line represents the information extracted from image I_1, while the bottom section, whose elements are primed, refers to the information from image I_2. The attribute columns are selfexplanatory. The entries in the relationship column are made by using the concept of consistency, which is defined as follows: Two vertices v_i and v_j with coordinates (x_1,y_1) and (x_2,y_2) are said to be consistent if

$$|f(x_1,y_1,x_2,y_2,k_1, \ldots, k_9)| < \theta \qquad (2.6\text{-}6)$$

where $|f|$ is the absolute value of f and θ is a positive threshold. Ideally, two matching vertices should yield $f = 0$. The use of a threshold is introduced to allow for distortions in imaging and computer processing.

A pair of consistent vertices in an image define a consistent edge. Two regions R_i and R_j are said to be K-consistent if K percent of their vertices are consistent. In theory, two truly matching regions should have the same number of vertices and every vertex in one region should have a consistent vertex in the other. In practice, the distortions mentioned above often lead to considering a match of K% of these vertices as the condition for consistency between two regions. Each of the remaining inconsistent vertices is then merged with the nearest vertex that was found to be consistent.

With reference to the last column in Table 2.8, and using the concepts just described, a region R_i in image I_1 is said to be related to a region R_j in I_2 if they are K-consistent. Use of the null set symbol, ϕ, indicates that R_i may not have a consistent region in I_2. The essence of Hwang's approach is to use a relaxation-labeling algorithm (Rosenfeld et al. [112]) with the above equation as a constraint to find consistent regions and merge those regions in an image that were found to be inconsistent. The final result of this procedure is a (usually reduced) relational table all of whose entries are consistent.

Elements (Regions)	Attributes: Vertices	Attributes: Edges	Relationship "Consistent"
R_1	$v_{11}, v_{12}, \ldots, v_{1m_1}$	$e_{11}, e_{12}, \ldots, e_{1p_1}$	R'_h or ϕ
R_2	$v_{21}, v_{22}, \ldots, v_{2m_2}$	$e_{21}, e_{22}, \ldots, e_{2p_2}$	R'_j or ϕ
$\vdots$	$\vdots$	$\vdots$	$\vdots$
R_M	$v_{M1}, v_{M2}, \ldots, v_{Mm_M}$	$e_{M1}, e_{M2}, \ldots, e_{Mp_M}$	R'_L or ϕ
R'_1	$v'_{11}, v'_{12}, \ldots, v'_{1n_1}$	$e'_{11}, e'_{12}, \ldots, e'_{1q_1}$	R_k or ϕ
R'_2	$v'_{21}, v'_{22}, \ldots, v'_{2n_2}$	$e'_{21}, e'_{22}, \ldots, e'_{2q_2}$	R_i or ϕ
$\vdots$	$\vdots$	$\vdots$	$\vdots$
R'_N	$v'_{N1}, v'_{N2}, \ldots, v'_{Nn_N}$	$e'_{N1}, e'_{N2}, \ldots, e'_{Nq_N}$	R_G or ϕ

Table 2.8. Relational table for object location.

Use of the approach described above is illustrated in Fig. 2.53. Parts (a) and (b) of this figure were obtained by digitizing the images produced by two cameras at different locations and viewing angles with respect to a three-dimensional solid. Figures 2.53(c) and (d) show the corresponding results after region extraction, and Figs. 2.53(e) and (f) are the vertices and edges defining the boundaries of the regions in each image. Figures 2.53(g) and (h) show the consistent regions obtained with $\theta = 0.707$ and $K = 95\%$. Note that the interior regions, because of their irregularities, failed to meet the consistency test. These two resulting views were then used to reconstruct the object in three-dimensional space. As an illustration of the accuracy of the reconstruction process, Figs. 2.53(i) and (j) show the reconstructed object projected back onto the original image planes. The importance of these results is that they were obtained without using a camera model. As mentioned earlier, once the three-dimensional coordinates of the object in question are known, they can be used to establish its true location with respect to other objects in the scene.

2.7 Examples of Systems for Digital Scene Analysis and Image Processing

We conclude this chapter with a brief summary of state-of-the-art scene analysis/image processing systems. For each system we list the design goals, the hardware-software configurations, and the database organizations reported by the designers.

2.7.1 Multisensor Image Database System (MIDAS)

This system was developed of a research tool for image processing and scene analysis to support research in the following areas:

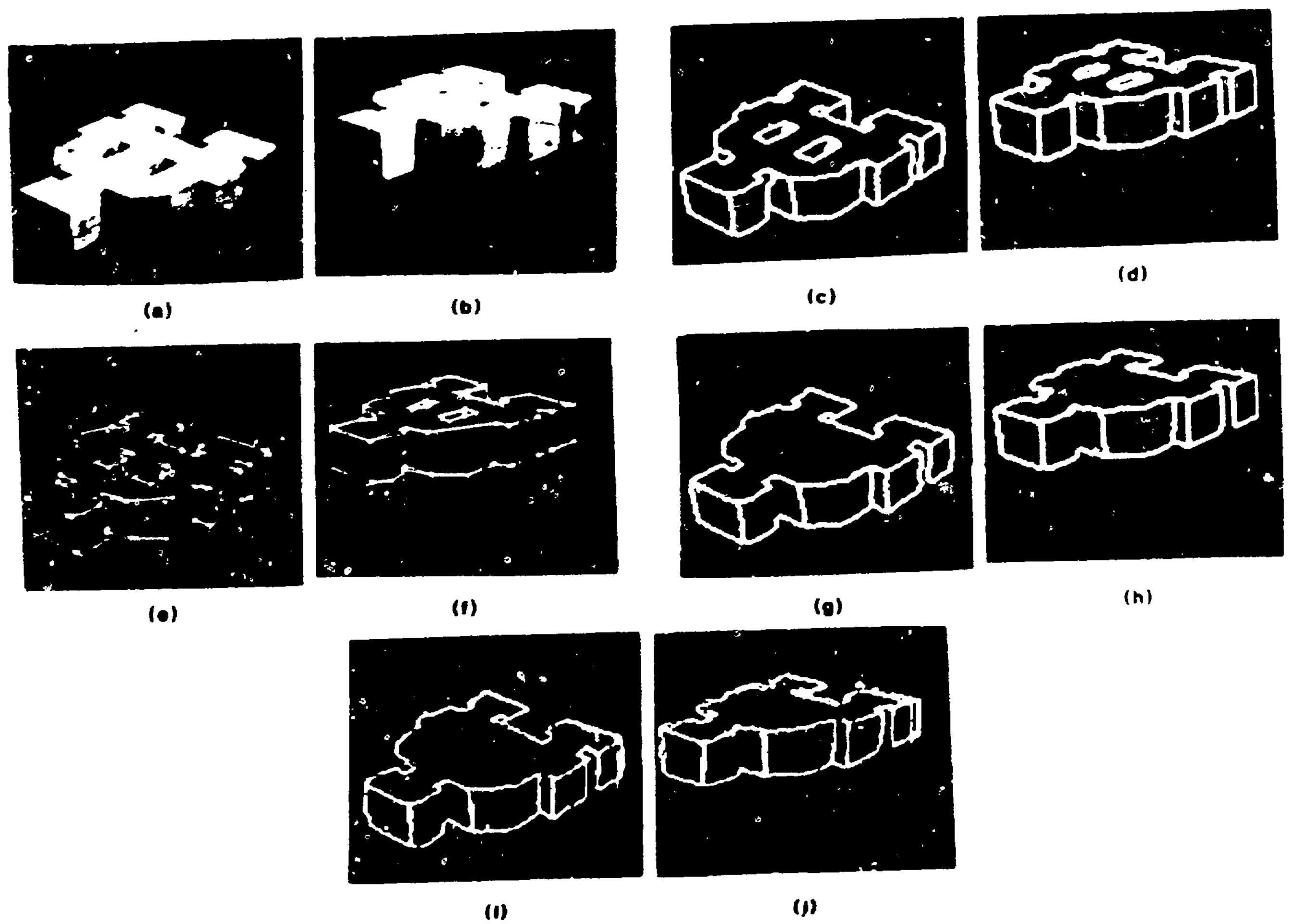


Figure 2.53. Two views of a three-dimensional object at various stages of processing.

(i) Performance evaluation of a variety of algorithms at various levels of image representation and abstraction; these levels include real digitized images and hierarchical symbolic descriptions of images.

(ii) Error analysis as an extension of performance evaluation to focus on the nature of erroneous results in automatic extraction of scene descriptions.

(iii) Development of knowledge acquisition techniques for the automated learning of structural and feature descriptions of objects and of symbolic feature primitives.

The MIDAS facility was designed to function in an environment in which there are several hundred images that range from LANDSAT multispectral pictures to electron photomicrographs [107]. The system permits multiple representation of scenes and offers a high degree of data independence in which users explicitly define relationships as needed.

A real image (called the "signal representation" of a scene) is maintained as a collection of reduced images, with the number of reductions by powers of 2 determined by the size of the original. Symbolic representations of an image are stored as relational DB's with object or subobject names attached to collections of real image picture points; the six levels of symbolic representation (scene type, scene, cluster, object, region, segment) form a hierarchical model of an image.

Hardware-Software: The MIDAS hardware organization is shown in Fig. 2.54. The multiprocessor configuration, using the UNIX operating system, is one factor in providing less than a one second response time to database requests.

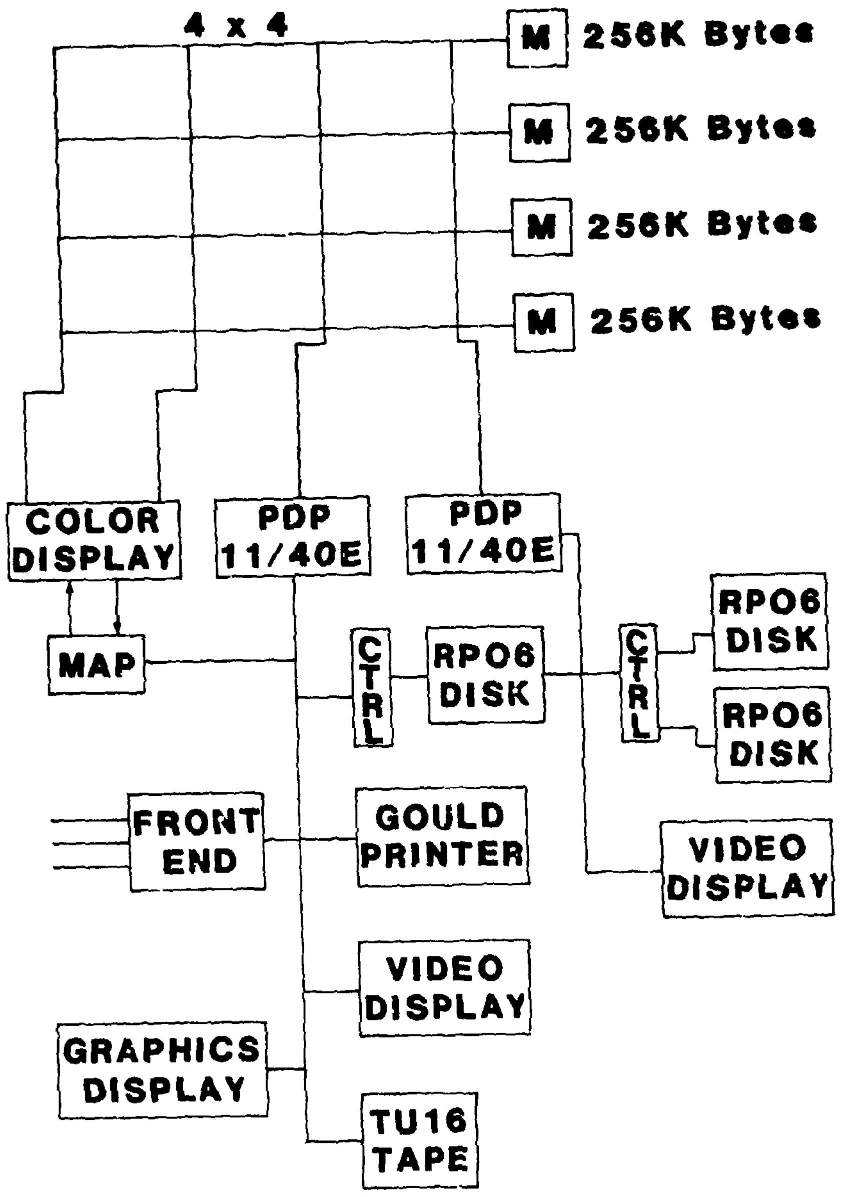


Figure 2.54. MIDAS hardware organization.

Interactive software packages provide MIDAS's image DB manipulation capabilities. The software incorporates memory paging to handle large images and provides access to a library of subroutines for image modification, analysis, and display. These subroutines are considered to be "picture processing primitives;" low level routines control buffer transfers, and higher level routines exist for operations such as histogram generation, extraction of subimages, and rescaling of pixel data.

DB Organization: For a given image, both the signal and the symbolic representations are hierarchies--the real image according to resolution (reduction) and the symbolic image according to six levels illustrated in Fig. 2.55.

The hierarchical symbolic DB is stored as an image description file (IDF) that contains the tree structure along with such general information as the person who generated the segmentation, the number of regions and their reference numbers, and each region's location given by the coordinates of a minimum bounding rectangle (MBR). A typical subset of an IDF appears to the right of the scene tree above. Symbolic representations generally are interactively formed. Vector lists of MBR's are also interactively defined to tie the symbolic representations into subimages in the signal representations.

The overall DB organization is as a text file containing a unique scene file for each image in MIDAS (Fig. 2.56). A scene file contains such data as the names of the signal representation files, pixel size, kind of sensor providing the original image, and the number of rows and columns. There are also pointers to IDF's that have been created for the image. There is provision for a relational DB to be generated

SCENE DESCRIPTION TREE

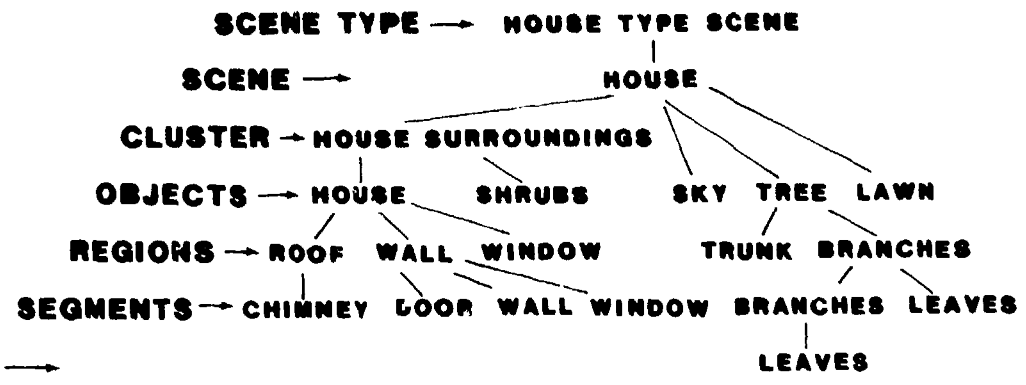

SCENE TYPE → HOUSE TYPE SCENE
SCENE → HOUSE
CLUSTER → HOUSE SURROUNDINGS
OBJECTS → HOUSE SHRUBS SKY TREE LAWN
REGIONS → ROOF WALL WINDOW TRUNK BRANCHES
SEGMENTS → CHIMNEY DOOR WALL WINDOW BRANCHES LEAVES
LEAVES

IMAGE DESCRIPTION FILE (IDF)

```
EXACT SEG(0) REGION IS HOUSE
REG    = 19    LEVEL = SCENE
MCKEOWN        ANCESTOR = -1
1-14-77        DESCENDANTS = 1234
HOUSE    = R0  MBR = 1,1:1,800:600,800:600,1
HOUSE SURROUNDINGS = R1
SKY=R2     (1) REGION IS HOUSE SURROUNDINGS
ROOF = R15     LEVEL = CLUSTER
WINDOW = R16   ANCESTOR = 0
               DESCENDANTS = 15 16 19
LEAVES=R18     MBR = 57,94:57,738:441,738:441,94
LEAVES=R19

          (16) REGION IS WINDOW
               LEVEL = REGIONS
               ANCESTOR = 5
               DESCENDANTS = -1
               MBR = 244,621:244,272:259,672:259,621
```

Figure 2.55. Sample tree and file organizations used in the MIDAS system.

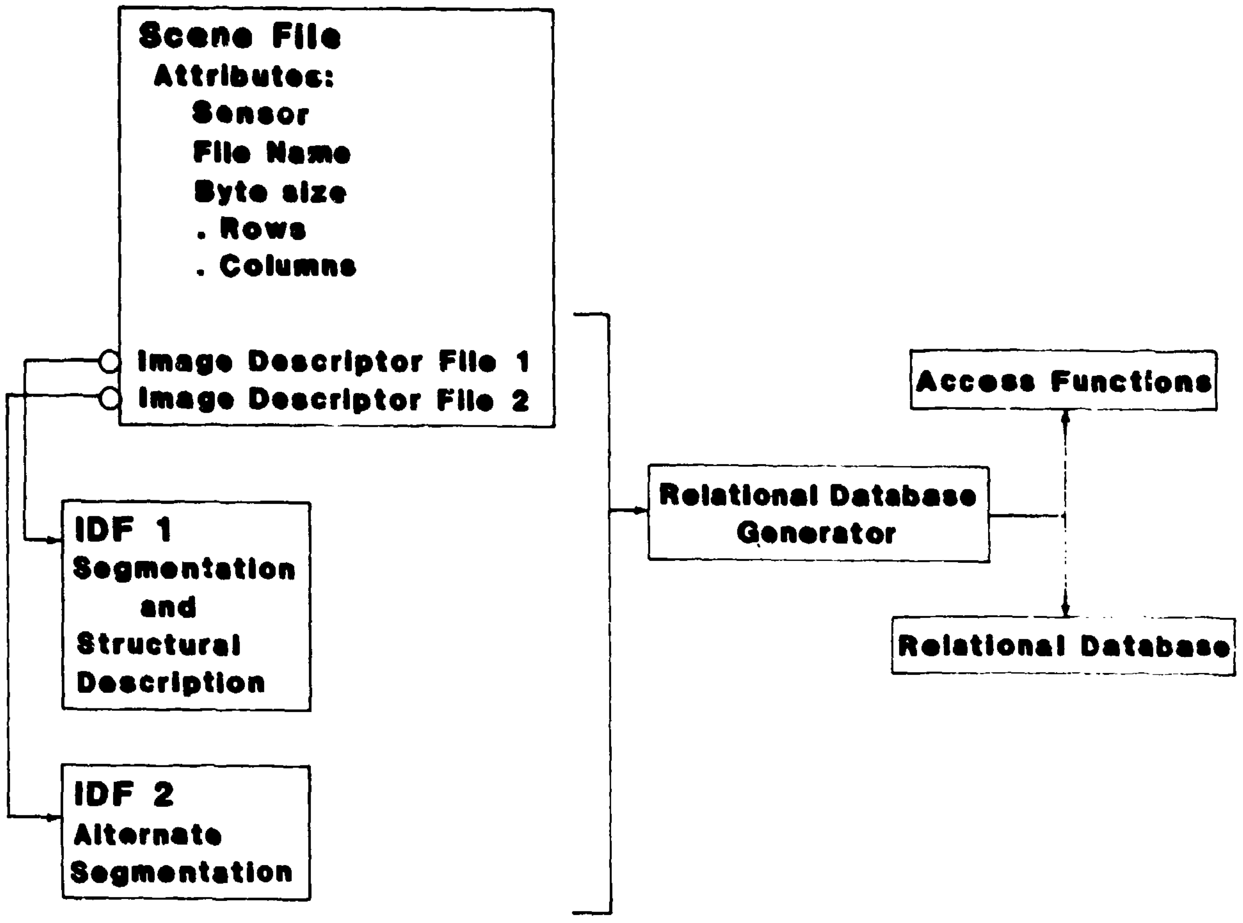


Figure 2.56. Scene file, IDF's, and relational DB diagram.

automatically for a scene file to afford rapid response to relation-based queries.

2.7.2 Relational Database System for Pictures

This system is based on a relational DB organization for complex pictures in which color and texture are used to establish region boundaries (Kunii et al. [102]). The DB organization satisfies the following requirements:

(i) Storage of character and numerical data in the same DB.

(ii) Data independence from the characteristics of specific computers.

(iii) Data independence from a restricted user's viewpoint, i.e., flexibility for a variety of users with different interpretations of picture data.

(iv) Freedom from unnecessary linkages in order to support incomplete data and disparate sources of data.

(v) Flexibility to allow data structures such as networks or graphs to be imposed on a copy of the picture data by a user.

The DBS was expected to be useful to users with different views of data collection and picture representation, i.e., users with different world models. (In general DBS design for commercial applications, a single dominant view of the "real world" is often the basis on which the data structures and DB's are selected; all users are assumed to agree on this model and to share the same concepts of usage of the data). No hardware-software details are given by Kunii et al. [102].

DB Organization: Picture data is organized as a set of relational DB's associated level by level with a hierarchy for symbolic picture representation (picture, object, part, region, color/texture).

The picture-level DB entries are picture reference numbers (SNAP#) with numerical attributes such as DATE and NEGATIVE# and character string attributes such as PLACE (taken) and SUBJECT. SNAP# is a primary key. An illustration of part of the object-level DB table for two SNAPs was shown in Fig. 2.48. A generic name is attached to each extracted object, and the SUPERPOSITION ORDER attribute is a nonnegative integer giving relative distance of an object's center of mass from the sensor that created the image. A candidate key for this figure is [SNAP#, OBJECT#] because each specification of these two attributes that is a legal combination for the relational table uniquely implies the values of the remaining attributes.

The part-level table represents objects as collections of parts. Each part is assigned a reference number, a generic name like LEAVES or LAWN, and a SUPERPOSITION ORDER number. Entries at the level of regions are connected sets of pixels having identical color (gray level, hue, saturation) and texture attributes in the original image. Regions have boundaries which can be transferred upwards to parts, objects, and pictures. Values for COLOR can be obtained via primary color filters. Boundaries can be described by a list of sample points with (x,y) coordinates or by parameterized curves fit to the data points. TEXTURE is also a region attribute.

2.7.3 Relational Pictorial Database System

This is an integrated pictorial DBS for efficient representation of physical images and logical pictures (Chang et al. [86, 87]). The physical images are digitized real images occupying a separate, paged, image store handled by an Image Store Management System (ISMS). The logical pictures are representations of the physical images as collections of picture objects stored as a set of relational DB's managed by a Relational Algebraic Interpreter (RAIN). Correspondences between physical images and logical pictures are established by a Picture Page Table (PPT), a relational DB that links objects in a logical picture to memory pages (which are essentially rectangular subimages)in a real image.

Hardware-Software: The pictorial analysis system is supported by a Graphics-Oriented Relational Algebraic Interpreter (GRAIN), as shown in Fig. 2.57. User interaction is accomplished via a touch-panel plasma display. The host machine is a PDP-11/40 with a UNIX operating system. System software is written in the C lnaguage, a high-level UNIX-supported language.

DB Organization: The ISMS interprets host commands for the image store processor. These commands are essentially independent of the physical image hardware, and some concurrent operations are possible. Two versions of an image are defined in the ISMS:

(i) A contour picture describes a line drawing for a logical picture object as a minimum enclosing rectangle (MER). Five formats for contour representation, including chain codes or (x,y) coordinates with a directory, are allowed.

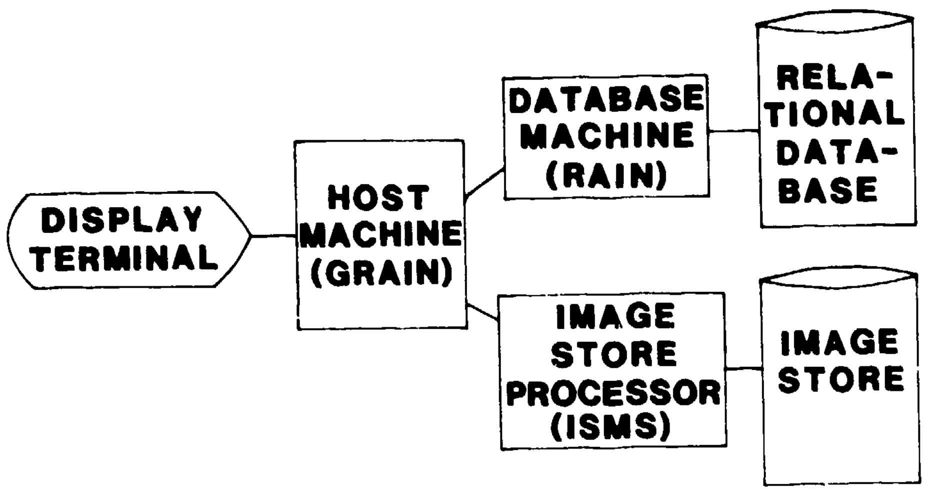


Figure 2.57. Basic elements of the GRAIN system.

(ii) A digitized image defines a rectangle of arbitrary size and orientation with respect to a cartesian coordinate system. Each pixel in a rectangle is addressed by its (x,y) coordinates and has an associated nonnegative integer value with gray-level information.

A picture object contained in a MER must be partitioned into rectangular picture pages to be accessed from the image store; the page size is not fixed but must not exceed an upper limit determined by memory buffer constraints. The paging is guided by a user-specified cost function which, for instance, could reflect that a minimum number of pages is desired or that minimum total area coverage by the rectangular subimages is desired.

A logical picture is represented as three relational DB's:

(i) The Picture Object Table (POT) has individual objects as its rows. The columns for an object row give a labeling position, the name of the picture object of which the row object is a component, a tilting angle to define the object's orientation with respect to the object of which it is a part, and a number of additional attributes as necessary to complete the characterization. In the position label, the entry (-1,-1) flags a composite object; otherwise, the object has a contour description in a contour table, PCT.

(ii) The Picture Contour Table (PCT) contains contour information about objects in the POT. The contour is defined by one of the formats shared with the image store contour picture and may reside in the image store itself or be a part of the PCT.

(iii) The PPT mentioned above specifies which of the image store picture pages correspond to a given object in logical picture.

Table 2.9 is an example of a POT for objects A, B, C, and D with the following hierarchy:

$$\begin{tikzcd}
 & X \arrow[dl, no head] \arrow[dr, no head] & & \\
C & & D \arrow[dl, no head] \arrow[dr, no head] & \\
 & A & & B.
\end{tikzcd}$$

The remaining columns for object A, for example, give (x - y-angle) positional information and specify that A possesses attribute A_1. Table 2.10 is an associated PCT in which object A is designated type 4 in contour format (which here means encoded as chain codes) and PIC.A is a reference name for a corresponding file as an image store contour table. Table 2.11 is an associated PPT showing, for example, that objects A and B each occupy one picture page whereas object C occupies two pages. Figure 2.58 is a representation of the logical picture showing where objects A, B, and C are located.

2.7.4 Image Database with Interactive Query Language

Important features of this system are an image storage format and an associated data manipulation language for storing, retrieving, and modifying large images primarily of the type produced by LANDSAT satellites [105, 106]. Emphasis was placed on the design of a high-level, interactive language called IQ (Image Query) to provide users with a

NAME	ONAME	LABEL		ANGLE	ATTRIBUTES			
		X	Y		A_1	A_2	A_3	A_4
A	D	0	4	0	1	0	0	0
B	D	4	4	0	0	1	1	1
C	X	1	1	0	0	0	1	1
D	X	-1	-1	0	0	0	0	0

Table 2.9. Picture object table (POT).

NAME	TYPE	UNAME	CONTOUR CODES
A	4	PIC.A	((0,4),(2,4)) ((2,4),(2,6)) ((2,6),(0,6)) ((0,6),(0,4))
B	4	PIC.B	((4,4),(6,4)) ((6,4),(6,6)) ((6,6),(4,6)) ((4,6),(4,4))
C	4	PIC.C	((1,1),(5,1)) ((5,1),(5,3)) ((5,3),(1,3)) ((1,3),(1,1))

Table 2.10. Picture contour table (PCT).

NAME	NPAGE	LOCATION		ANGLE	PAGE SIZE	
		X_p	Y_p		l_x	l_y
A	0	0	4	0	2	2
A	1	0	4	0	2	2
B	0	4	4	0	2	2
B	1	4	4	0	2	2
C	0	1	1	0	4	2
C	1	1	1	0	2	2
C	2	3	1	0	2	2

Table 2.11. Picture page table (PPT).

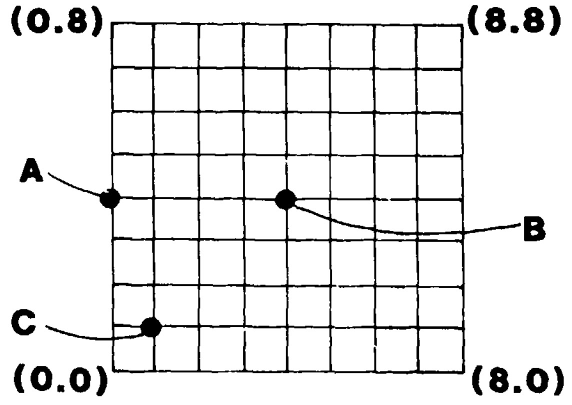


Figure 2.58. Location of objects in a logical picture.

variety of operators to be applied to images via statements with relatively simple syntax. This Image Database System (IDBS) does not support symbolic or logical representations of images per se; rather, users can process groups of pixels by operations such as expanding or compressing the range of pixel values, expanding or compressing the size of images, coloring images, joining a secondary image to a dominant image, and masking parts of an image. The IDBS was designed for image processing tasks requiring fast transfers of large blocks of data in secondary storage.

Hardware-Software: The IMBD system is implemented at the Data System Laboratory of the NASA Marshall Space Flight Center on a PDP-11/45 with 128K bytes of main memory and 600M bytes of disk storage. The operating system is RSX-11D. Peripherals include the usual I/O devices as well as color graphics and graphics input with a cursor.

Users employ the query language IQ to manipulate images. Windows, which are polygons that are interpreted either to enclose or to exclude subimages, are created by users as artifacts for the IQ functions; in other words, there are operations on images and windows to produce new images and windows. The following five operations are included in the IQ language:

(1) JOIN (join an ordered pair of images to create a new image).

(2) MASK (mask a subimage by a window, either as an enclosure mask or as an exclusion mask).

(3) CLIP (mask a subimage by a window, then discard the unmasked subimage).

(4) UNION (join an ordered pair of windows to create a new one).

(5) INTERSECTION (intersect an ordered pair of windows to create a new one).

Four functions are "generic" operations in the sense that parameters must be preset to create specific actions:

(6) TRANSFORM (linearly transform pixel values into a new range).

(7) COLOR (attach color to a range of pixel values).

(8) OVERLAY (create a new image by overlaying one on another).

(9) ZOOM (expand or compress the size of an image).

IQ statements are also provided for a user to create functions, to route images and display devices, and to compute and exhibit joint histograms and joint pixel value distributions.

DB Organization: An image is partitioned into subimage pages of 64 by 64 pixels and has an associated page table that provides the disk address of each page. A system file directory is maintained with the following information about each permanent image: name, disk address of its page table, size, category (such as soil type, slope, or land-ownership) for a map or spectral channel (such as radar or infrared) for a satellite image, geographical coordinates, scale factor, and other explanatory items. This internal organization is the basis for the DBMS operations and is transparent to the user. A user can create temporary files of processed images and can also create windows as sets of polygons represented by lists of vertices in longitude/latitude.

2.7.5 Partititoned Large Image System

This is an image processing facility with a DBMS for manipulating images with large numbers of pixels (up to 4000 by 4000). The image files use the Standard Image Database (SIDBA) for Japan, a set of standard formats for storing image data on media such as magnetic tape. The SIDBA is an effort to develop a specification which has enough flexibility to make it acceptable to users at different locations and which can be used for development of transportable software (Tamura et al. [116, 117]).

Hardware-Software: Figure 2.59 shows the image processing facility that uses a TOSBAC 40C minicomputer as a I/O controller to connect its peripherals to a large mainframe, the TOSBAC 5600/160. The software consists of a library of image routines and a supervisor that interprets a user's call, accesses the required image in storage, and delivers the data to the specified routines. If an image is too large for all the data to reside in main memory at once, the supervisor handles the sequential transferring of subimages.

The algorithms in the library are FORTRAN subroutines for two-dimensional array data and include, but are not restricted to, several universal utility routines for SIDBA images. As examples, a user can invoke utilities to change gray level information, to list the names and sizes of images in a file, or to change subimage size.

DB Organization: Real images are represented in the SIDBA format as a collection of image files. For the storage of images on a magnetic tape, for example, the SIDBAMT specification is for a header of 512

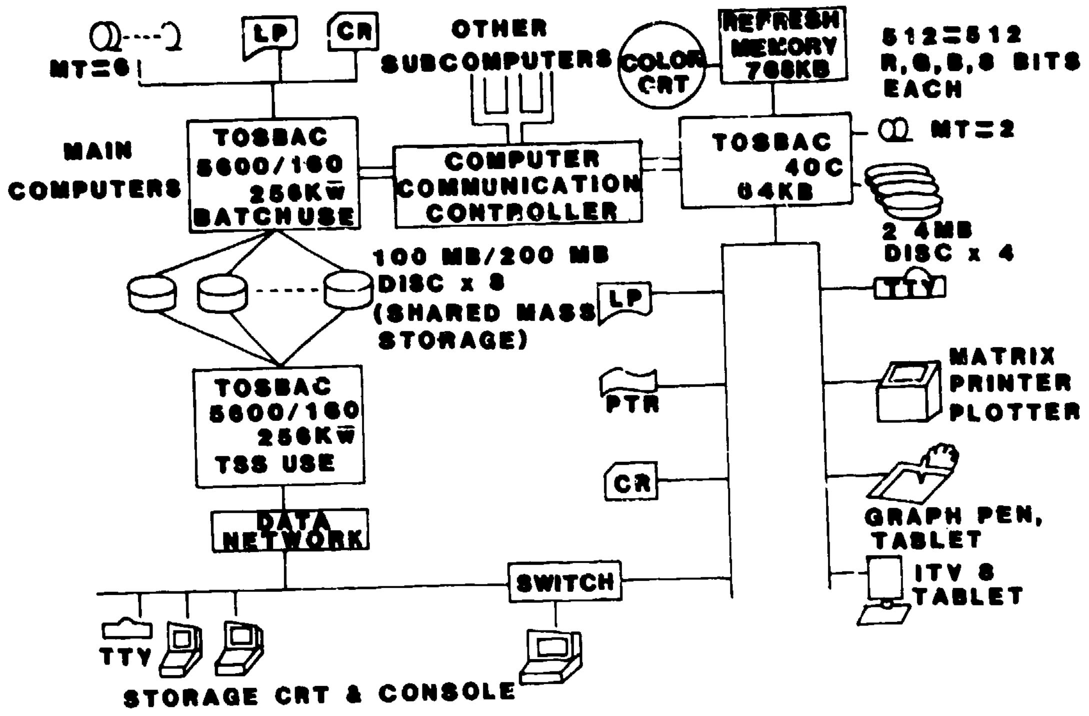


Figure 2.59. Electrotechnical laboratory image processing facility.

No.	Items	Length (bytes)	Position	Symbols
1	Data Name	12*	1-12	DNAME
2	Frame Size	4	13-16	FX, FY
3	Subframe Size	4	17-20	SX, SY
4	Number of Subframes	4	21-24	NSX, NSY
5	Gray level Information			
	Assigned Bits	1	25	NBIT
	Effective Bits	1	26	EBIT
	Storing Form	1	27	LR
	Gray Scale Type	1	28	SCALE
6	Logical Record Length	4	29-32	LLENG
7	Physical Record Length	2	33-34	PLENG
8	Pixel Order	1	35	PODR
9	Subframe Order	1	36	SODR
10	Margin Width	4	37-40	ML, MR, MU, MD
11	Input Device	6*	41-46	
12	Picture Type	6*	47-52	
13	Size of Fully Visible Area	4	53-56	VX, VY
14	Coordinates of Reliable Area	8	57-64	RXS, RYS, RXE, RYE
15	Starting Point of Frame	4	65-68	FXS, FYS
16	Date	6*	69-74	
17	Location	6*	75-80	
18	Reserved for Future Use	52	81-132	
19	Comments	380*	133-512	
	Total	512	*: Character Code (EBCDIC)	

Table 2.12. Format of the header in SIDBAMT.

bytes for each image with the header fields shown in Table 2.12. Item 2 is the image (frame) size; items 3 and 4, the size and number of subimges. Gray-level information on item 5 includes the total number of bits assigned for the levels and the effective number of bits for the actual quantization. In item 6, each subframe is a logical record; in item 7, the maximum physical record length is set at 4096 bytes. The remainder of the header contains other aspects of the physical DB.

REFERENCES

1. T. Okoshi, "Three-Dimensional Imaging Techniques," Academic Press, NY, 1976.

2. R. O. Duda, D. Nitzan, and P. Barrett, "Use of Range and Reflectance Data to Find Planar Surface Regions " IEEE Trans. Pattern Anal. Machine Intel., vol PAMI-1, no. 3, pp. 259-271, 1979.

3. R. C. Gonzalez and P. Wintz, Digital Image Processing, Addison-Wesley, Reading,MA, 1977.

4. J. S. Lee, "Digital Image Enhancement and Noise Filtering by Use of Local Statistics," IEEE Trans. Pattern Analysis and Machine Intell. vol. PAMI 2, no. 2, pp. 165-168, 1980.

5. J. S. Lee, "Refined Filtering of Image Noise Using Local Statistics," Report 8374, January, 1980,Naval Research Laboratory, Washington, D.C.

6. R. C. Gonzalez and P. Wintz, Digital Image Processing, Addison-Wesley, Reading, MA, 1977.

7. R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," Trans. ASME J. Basic Eng., vol. 82, pp. 35-45, 1960.

8. A. K. Jain, "A Semicausal Model for Recursive Filtering of Two-Dimensional Images," IEEE Trans. Comput., vol. C-12, pp. 734-738, 1972.

9. A. E Bryson,Jr. and Y. C. Ho, Applied Optimal Control, Chapter 12, Blaisdell Publishing Co., Waltham, MA, 1969.

10. J. M. Lloyd, Thermal Imaging Systems, Plenum Press, NY, 1975.

11. R. C. Gonzalez and B. A. Fittes, "Gray-Level Transformations for Interactive Image Enhancement," J. Mechanism and Machine Theory, vol. 12, pp. 111-122, 1977.

12. S. K. Chang and Y. W. Wong, "Optimal Histogram Matching by Monotone Gray-Level Transformation," Comm. ACM, vol. 21, pp. 835-840, 1978.

13. R. A. Hummel, "Histogram Modification Techniques," CGIP, vol. 4, pp. 209-224, 1975.

14. R. A. Hummel, "Image Enhancement by Histogram Transformation," CGIP, vol. 6, pp. 184-195, 1977.

15. R. Y. Wong, E. L. Hall,and J. D. Rouge, "Image Intensity Transformations," Proc. IEEE Comp. Soc. Conf. on Pattern Recog. and Image Processing, pp. 96-99, 1978.

16. W. Frei, "Image Enhancement by Histogram Hyperbolizations," CGIP, vol. 6, pp. 184-195, 1977.

17. R. E. Woods and R. C. Gonzalez, "Real-Time Digital Image Enhancement," Proceedings of the IEEE, vol. 69, no. 5, May, 1981.

18. A. Rosenfeld, "Progress in Picture Processing: 1969-71," Computing Surveys, vol. 5, pp. 81-108, 1973.

19. A. Rosenfeld, "Picture Processing," Comput. Graph. and Image Proc., vols. 1-7, 1972-1978.

20. K. S. Fu and J. K. Mui, "A Survey of Image Segmentation," Pattern Recognition, vol. 13, pp. 3-16, 1981.

21. J. S. Weska, "A Survey of Threshold Selection Techniques," Comput. Graph. Image Proc., vol. 7, pp. 259-265, 1978.

22. R. B. Ohlander, "Analysis of Natural Scenes," Ph.D. Dissertation, Department of Computer Science, Carnegie-Mellon University, Pittsburg, Penn., 1975.

23. S W. Zucker, "Region Growing: Childhood and Adolescence," Comput. Graph. Image Proc., vol. 5, pp. 382-399, 1976.

24. T. Pavlidis, Structural Pattern Recognition, Springer-Verlag, New York, 1977.

25. A. Rosenfeld, R. Hummel,and S. W. Zucker, "Scene Labeling by Relaxation Operations," IEEE Trans. Syst. Man & Cyb., vol SMC-6, pp. 420-433, 1976.

26. M. Kelly, "Visual Identification of People by Computer," Stanford University, Artificial Intelligence Project Memo. 121, 1970.

27. C. A. Harlow and S. A. Eisenbeis, "The Analysis of Radiographic Images," IEEE Trans. Comput., vol. C-22, pp. 678-689, 1973.

28. M. Yachida, M. Ikeda, and S. Tsuji, "A Plan-Guided Analysis of Cineangiograms for Measurement of Dynamic Behavior of Heart Wall," IEEE Trans. Patt. Anal. and Machine Intell., vol. PAMI 2, pp. 537-543, 1980.

29. J. M. Tenenbaum and H. G. Barrow, "IGS: A Paradigm for Integrating Image Segmentation and Interpretation," Proc. Third Int. Joint Conf. Pattern Recog., pp. 504-513, 1976.

30. A. R. Hansen and E. M. Riseman, "Segmentation of Natural Scenes," Computer Vision Systems, Academic Press, NY, 1978.

31. E. L. Hall, Computer Image Processing and Recognition, Academic Press, NY, 1979.

32. J. M. S. Prewitt, "Object Enhancement and Extraction," In Picture Processing and Psychopictorics, edited by B. S. Lipkin and A. Rosenfeld, Academic Press, NY, 1970.

33. R. Ohlander, K. Price, and D. R. Reddy, "Picture Segmentation Using a Recursive Region Splitting Method," Computer Graphics and Image Processing, vol. 8, pp. 313-333, 1978.

34. C. R. Brice and C. L. Fennema, "Scene Analysis Using Regions," Artificial Intelligence, vol. 1, pp. 205-226, 1970.

35. D. J. Bryant, "Evaluation of Edge Operators Using Relative and Absolute Grading," Thesis, Electrical Engineering, the University of Tennessee, Knoxville, December, 1978.

36. W. K. Pratt, Digital Image Processing, John Wiley & Sons, NY, 1978.

37. I. Munro, "Efficient Determination of the Transitive Closure of a Directed Graph," Information Processing Letters, vol. 1, no. 2, pg. 56, 1971.

38. A. Sage, Methodology for Large Scale Systems, McGraw-Hill, NY, 1977.

39. E. L. Hall and R. C. Gonzalez, "Scene Content Analysis, Measurement, Refinement and Verification," Technical Report, SAMSO-TR-79-46, The Space and Missile System Organization, December, 1978.

40. E. L. Hall, J. J. Hwang, and M. Hwang Sadjadi, "Scene Content Analysis for the Autonomous Terminal Homing Program," Technical Report, BMO-TR-79, November, 1979.

41. G. S. Robinson, "Edge Detection by Compass Gradient Masks," Computer Graphics and Image Processing, vol. 6, pp. 492-501, 1977.

42. E. L. Hall and J. J. Hang, Monthly Report, Scene Content Analysis Program, SAMSO F04701-78-C-0193, March, 1979.

43. A. Scher, F. R. D. Velasco, and A. Rosenfeld, "Some New Image Smoothing Techniques," IEEE Trans. SMC, vol. SMC-10, no. 3, pp. 153-158, March, 1980.

44. J. O. Eklundh and A. Rosenfeld, "Image Smoothing Based on Neighbor Linking," Technical Report TR-773, Department of Computer Science, University of Maryland, June, 1979.

45. T. Pavlidis, "A Review of Algorithms for Shape Analysis," Comput. Graphics Image Processing, vol. 7, pp. 243-258, 1978.

46. T. Pavlidis, "Algorithms for Shape Analysis of Contours and Wave-forms," IEEE Trans. PAMI, vol. PAMI-2, no. 4, pp. 301-312, 1980.

47. A. Rosenfeld and A. C. Kak, Digital Picture Processing, Academic Press, NY, 1976.

48. F. A. Sadjadi and E. L. Hall, "Three-Dimensional Moment Invariants," IEEE Trans. PAMI, vol. PAMI-2, no. 2, pp. 127-136, 1980.

49. W. E. Snyder and D. A. Tang, "Finding the Extrema of a Region," IEEE Trans. PAMI, vol. PAMI-2, no. 3, pp. 266-269.

50. T. Pavlidis, Structural Pattern Recognition, Springer-Verlag, NY, 1977.

51. H. Freeman, "On the Encoding of Arbitrary Geometric Configurations," IEEE Trans. Elec. Comput., vol. EC-10, pp. 260-268, 1961.

52. H. Freeman and R. Shapira, "Determining the Minimum Area Encasing Rectangle for an Arbitrary Closed Curve," Comm. Assoc. Comput. Mach. vol. 17, no. 7, pp. 409-413, 1975.

53. L. G. Shapiro, "A Structural Model of Shape," IEEE Trans. PAMI, vol. PAMI-2, no. 2, pp. 111-126, 1980.

54. L. S. Davis, "Shape Matching Using Relaxation Techniques," IEEE Trans. PAMI, vol. PAMI-1, no. 1, pp. 60-72, 1979.

55. J. M. Prager, "Extracting and Labeling Boundary Segments in Natural Scenes, IEEE Trans. PAMI, vol. PAMI-2, no. 1, pp. 16-27, 1980.

56. K. C. You and K. S. Fu, "A Syntactic Approach to Shape Recognition Using Attributed Grammars," IEEE Trans. SMC, vol. SMC-9, pp. 334-344, 1979.

57. K. S. Fu, Syntatic Methods in Pattern Recognition, Academic Press, New York, NY, 1974.

58. R. C. Gonzalez and M. G. Thomason, Syntatic Pattern Recognition: An Introduction, Addison-Wesley, Reading, MA, 1978.

59. R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis, Wiley, NY, 1973.

60. E. Persoon and K. S. Fu, "Shape Descrimination Using Fourier Descriptors," IEEE Trans. Systems, Man, Cybernetics, vol. 7, pp. 170-179, 1977.

61. F. L. Alt, "Digital Pattern Recognition by Moments," _J. ACM_, vol. 9, pp. 240-258, 1962.

62. A. Papoulis, _Probability, Random Variables, and Stochastic Processes_, McGraw-Hill, NY, 1965.

63. M. Cohen and G. T. Toussaint, "On the Detection of Structures in Noisy Pictures," _Pattern Recognition_, vol. 9, pp. 95-98, 1977.

64. L. S. Davis, "Understanding Shape, II: Symmetry," _IEEE Trans. Systems, Man, Cybernetics_, vol. 7, pp. 204-212, 1977.

65. L. S. Davis, "Understanding Shape: Angles and Sides," _IEEE Trans. Computers_, vol. 26, pp. 236-242, 1977.

66. H. Freeman and L. S. Davis, "A Corner-Finding Algorithm for Chain-Coded Curves, _IEEE Trans. Computers_, vol. 26, pp. 297-303, 1977.

67. H. Freeman, "Shape Description Via the Use of Critical Points," _Proceedings, IEEE Computer Society Conference on Pattern Recognition and Image Processing_ (June 6-8, 1977, Troy, NY), IEEE Publ. 77CH 1208-9 C, IEEE Computer Society, Long Beach, CA, pp. 168-174, 1977.

68. A. K. Agrawala and A. V. Kulkarni, "A Sequential Approach to the Extraction of Shape Features," _Computer Graphics Image Processing_, vol. 6, pp. 538-557, 1977.

69. T. C. Woo, "Progress in Shape Modeling," (Chien, R. T. guest ed.,) Productivity and Automation, _Computer_, vol. 10(12), Dec., pp. 40-46, 1977.

70. M. S. Bartlett, "An Introduction to the Analysis of Spatial Patterns," _Suppl. Adv. Applied Probability_, vol. 10, pp. 1-13 (as well as many other papers in the same volume), 1978.

71. G. H. Granlund, "In Search of a General Picture Processing Operator," _CGIP_, vol. 8, pp. 155-173, 1978.

72. G. Winkler and K. Vattrodt, "Measures for Conspicuousness," _CGIP_, vol. 8, 1978.

73. T. Pavlidis and F. Ali, "A Hierarchical Syntactic Shape Analyzer," _IEEE Trans. Pattern Anal. Machine Intelligence_, vol. PAMI-1, pp. 2-9, 1979.

74. J. T. Tou and R. C. Gonzalez, _Pattern Recognition Principles_, Addison-Wesley, Reading, MA, 1974.

75. T. Pavlidis, "Linear and Context-Free Graph Grammars," _J. ACM_, vol. 19, no. 1, pp. 11-22, 1972.

76. A. Rosenfeld and D. L. Milgram, "Web Automata and Web Grammars," in Machine Intelligence-7, (B. Meltzer and D. Michie, eds.) Wiley, NY, 1972.

77. J. Feder, "Plex Languages," Inform. Sci., vol. 3, pp. 225-241, 1971.

78. A. W. Biermann and J. A. Feldman, "On the Synthesis of Finite-State Machines from Samples of their Behavior," IEEE Trans. Comput., vol. C-21, no. 6, pp. 592-597, 1972.

79. R. C. Gonzalez, J. J. Edwards, and M. G. Thomason, "An Algorithm for the Inference of Tree Grammars," Int. J. Comput. Inform. Sci., vol. 5, no. 2, pp. 145-164, 1976.

80. M. G. Thomason and R. C. Gonzalez, "Syntactic Recognition of Imperfectly Specified Patterns," IEEE Trans. Comput., vol. C-24, pp. 93-95, 1975.

81. K.S. Fu, "Recent Developments in Pattern Recognition," IEEE Trans. Comput., vol. C-29, no. 10, pp. 845-854, 1980.

82. M. Minsky, "A Framework for Representing Knowledge," in The Psychology of Computer Vision, P. H. Winston, ed., McGraw-Hill, NY, 1975.

83. A. V. Aho, J. E. Hopcroft, and J. D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1974.

84. A. V. Aho and J. D. Ullman, The Theory of Parsing, Translation, and Compiling, vol. 1, Prentice-Hall, Englewood Cliffs, NJ, 1972.

85. C. R. Brice and C. L. Fennema, "Scene Analysis Using Regions," Art. Intell., vol. 1, pp. 205-226, 1970.

86. S. K. Chang, N. Donato, J. Reuss, R. Rocchetti, and B. H. McCormick, "An Integrated Relational Database System for Pictures," Proc. IEEE Workshop on Picture Data Description and Management, pp. 142-149, April, 1977.

87. S. K. Chang, J. Reuss, and B. H. McCormick, "Design Considerations of a Pictorial Database System," Policy Anal. and Info. Sys., vol. 1, no. 2, pp. 49-70, 1978.

88. Codd, E. F., "A Relational Model of Data for Large Shared Data Banks," CACM, vol. 13, no. 6, pp. 377-387, June, 1970.

89. A. Klinger, K. S. Fu, and T. L. Kunii, editors, Data Structures Computer Graphics and Pattern Recognition, Academic Press, NY, 1977.

90. C. J. Date, An Introduction to Database Systems, Addison-Wesley, Reading, MA, 1977.

91. W. K. Giloi, Interactive Computer Graphics: Data Structures, Algorithms, Languages, Prentice Hall, Englewood Cliffs, NJ, 1978.

92. S. Gorn, "Explicit Definitions and Linguistic Dominoes," in Systems and Computer Science (J. F. Hart and S. Takusu, Eds.) Univ.of Toronto Press, Toronto, Canada, 1967.

93. E. Horowitz and S. Sahni, Fundamentals of Data Structures, Computer Science Press, Potomac, MD, 1977.

94. S. L. Horowitz and T. Pavlidis, "Picture Segmentation by a Tree Traversal Algorithm," J.ACM, vol. 23, no. 2, pp. 368-388, April, 1976.

95. G. M. Hunter and K. Steiglitz, "Operations on Images Using Quad Trees," IEEE Trans. Pat. Anal. Mach. Intell., vol. PAMI-1, no. 2, pp. 145-153, April, 1979.

96. J. J. Hwang, "Computer Stereo Vision for Three-Dimensional Object Location," Ph.D. Dissertation, Electrical Engineering Department, Univ. of Tenn., Knoxville, 1980.

97. J. J. Hwang, C. C. Lee, and E. L. Hall, "Segmentation of Solid Objects Using Global Local Edge Concidence," Proc. IEEE Conf. on Pat. Recogn. and Image Proc., pp. 114-121, August, 1979.

98. W. Kim, "Relational Database Systems," Comp. Surveys, vol. 11, no. 3, pp. 185-211, Sept., 1979.

99. A. Klinger and C. R. Dyer, "Experiments on Picture Representation Using Regular Decomposition," Comp. Graphics and Image Proc., vol. 5, pp. 68-105, March, 1976.

100. A. Klinger, M. L. Rhodes, and V. T. To, "Accessing Image Data," Policy Anal. and Info. Sci., vol. 1, no. 2, pp. 171-190, Jan., 1978.

101. D. E. Knuth, The Art of Computer Programming, Vol. 1, Fundamental Algorithms, Addison-Wesley, Reading, MA, 1968.

102. T. Kunii, S. Weyl, and J. M. Tenenbaum, "A Relational Database Scheme for Describing Complex Pictures with Color and Textures," Policy Anal. and Info. Sys., vol. 1, no. 2, pp. 127-142, 1978.

103. C. C. Lee, "Boundary Adjacency Tracing and Its Application to Database Compression," M.S. Thesis, Computer Science Dept., Univ. of Tenn, Knoxville, 1979.

104. C. C. Lee, J. H. Hwang, E. L. Hall, and M. G. Thomason, "Boundary Direction Vector Coding for Scene Description," Proc. IEEE Region 3 Conf., pp. 95-102, April, 1980.

105. Y. E. Lien and R. Schroff, "An Interactive Query Language for an Image Database," Policy Anal. and Info. Sys., vol. 1, no. 2, pp. 91-112, 1978.

106. Y. E. Lien and D. F. Utter, Jr., "Design of an Image Database," Proc. IEEE Workshop on Picture Data Description and Management, pp. 131-136, April, 1977.

107. D. McKeoun, Jr. and R. Reddy, "A Hierarchical Symbolic Representation for an Image Database," Proc. IEEE Workshop on Picture Data Description and Management, pp. 40-44, April, 1977.

108. W. M. Newman and R. F. Sproul, Principles of Interactive Computer Graphics, McGraw-Hill, New York, NY, 1979.

109. F. Palermo and D. Weller, "Picture Building System," Proc. IEEE Spring Compcon, San Francisco, pp. 235-237, 1979.

110. T. Pavlidis, "A Minimum Storage Boundary Tracing Algorithm and Its Applications to Automatic Inspection," IEEE Trans. SMC, vol. SMC-8, no. 1, pp. 66-69, January, 1978.

111. A. Rosenfeld, Picture Languages: Formal Models for Picture Recognition, Academic Press, NY, 1979.

112. A. Rosenfeld, R. A. Hummel, and S. Zucker, "Scene Labeling by Relaxation Operations," IEEE Trans. Syst. Man. Cyb., vol. SMC-6, pp. 420-433, 1976.

113. M. E. Senko, "Data Structures and Data Accessing in Database Systems, Past, Present, Future," IBM Sys. Jour., vol. 16, no. 3, pp. 208-257, 1977.

114. L. G. Shapiro, "Data Structures for Picture Processing: A Survey," Comp. Graphics and Image Proc., vol. 11, pp. 162-184, 1979.

115. Special Section on Pattern Recognition Software, IEEE Trans. Software Engrg., vol. SE-3, no. 2, pp. 160-190, March, 1977.

116. H. Tamura and S. Mori, "A Data Management System for Manipulating Large Images," Proc. IEEE Workshop on Picture Data Description and Management, pp. 45-54, April, 1977.

117. H. Tamura, S. Mori, and T. Shimada, "Data Management for Manipulating Partitioned Large Images," Policy Anal. and Info. Sys., vol. 1, no. 2, pp. 143-170, 1978.

118. S. L. Tanimoto, "Pictorial Feature Distortion in a Pyramid," Comp. Graphics and Images Proc., vol. 5, no. 3, pp. 333-352, Sept., 1976.

119. J. D. Ullman, Principles of Database Systems, Computer Science Press, Potomac, MD, 1980.

120. D. Weller and F. Palermo, "Database Requirements for Graphics," Proc. IEEE Spring Compcon, San Francisco, pp. 231-233, 1979.

121. Y. Yakimovsky and R. Cunningham, "A System for Extracting Three-Dimensional Measurements from a Stereo Pair of TV Cameras," Comput. Graph. Image Proc., vol. 7, pp. 195-210, 1978.

122. B. Widrow, "The Rubber-Mask Technique, Parts I and II," Pattern Recognition, vol. 5, no. 3, pp. 175-211, 1973.

123. E. Bribiesca and A. Guzman, "How to Descripe Pure Form and How to Measure Differences in Shape Using Shape Numbers," Pattern Recognition, vol. 12, no. 2, pp. 101-112, 1980.

124. E. Bribiesca, "Arithmetic Operations Among Shapes Using Shape Numbers," Pattern Recognition, vol. 13, no. 2, pp. 123-137, 1981.

3. COMPUTER SCENE SYNTHESIS

Scene synthesis algorithms are considered in this section. Rather than an exhaustive survey of all synthesis techniques, only key important and computationally difficult algorithms are considered. Since "scene" implies a three dimensional nature, a key problem is the three dimensional representation in the computer of sufficient data to produce a realistic image from the scene.

An interesting argument related to realistic scene representation has been described by Carl Sagan in his book, Broca's Brain (p. 17). He argues that we do not have the storage capacity in either our largest computers or in the human brain to understand a barely visible grain of salt. Salt, or sodium chloride, NaCl, consists of sodium, whose atomic weight is 22.99, and chloride, whose atomic weight is 35.45. Therefore, the molecular weight of NaCl is 58.44. One mole or molecular weight of NaCl weighs 58.44 grams and contains 6.02 x 10^{23} molecules. A barely visible grain of NaCl is approximately one microgram. A division gives the number of molecules in a microgram of salt as approximately 10^{16}. If we wish to know the three dimensional (x,y,z) position of each molecule, then a storage capacity of 3 x 10^{16} numbers is required. Sagan estimates that there are about 10^{11} neurons in the brain, each with perhaps 10^{3} dendrite connections. Thus, the total storage capacity of a human brain is about 10^{14}, less than 1% of the required storage, but much greater than any computer storage available. Sagan also points out that, if the salt is in an absolutely pure crystal, the position of every atom could be specified with only 10 bits of information.

This argument clearly illustrates the impossibility with modern technology to accurately represent a grain of salt, which is much simpler than an ordinary three dimensional scene. However, the tremendous benefit which could be obtained from observing regular structures or perhaps modelling irregularity is also strikingly clear.

In this section, the problems of three dimensional surface representation and display are considered in detail. First, surface representation is presented in Section 3.1. Next, the hidden surface problem which must be solved at TV rates to provide interactive scene viewing is described in Section 3.5. Surface shading computations are then considered in Section 3.6. Finally, the synthesis of irregular texture patterns is described in Section 3.7.

3.1 Curved Surface Representation

The representation of curved surfaces is an important topic in realistic scene synthesis and analysis. In a natural scene, the surface characteristics vary from point to point. This variation often cannot be modelled by planar or faceted surfaces. Therefore, a study of curved surfaces is essential for the synthesis and analysis of realistic scenes.

The problem of curve and surface representation is surprisingly complex. For example, in mathematical analysis, the problem of defining a curve is still a research topic. Recent papers on "snowflake" curves or fractals demonstrate curves which are continuous but nowhere differentiable, and illustrate that our common notion of dimension can be misleading by presenting a curve that goes through each point in an area. Even the concept of continuity is questioned when we consider a function composed of rational numbers. These counter-examples in analysis will not be considered relevant to the present study by restricting consideration to common methods used in computer scene synthesis.

Blinn [1] defines three common methods used to mathematically describe shapes. These are:

1. algebraic method in which all points on the surface satisfy an equation of the form

$$f(x,y,z) = 0 \;;$$

2. point set method in which all points on the surface or on the boundary edges of the surface are enumerated;

3. parametric method in which a surface is traced out by three bivariate functions

$$x(u,v),\ y(u,v),\ \text{and}\ z(u,v)$$

A set of operations which are required for modelling and surface synthesis are also listed by Blinn. These consist of:

1. Definition operations required for specifying the objects in a scene;
2. Basic geometric transformations including translations, rotations, sclaings, and perspective projections;
3. Calculation of intersections with other objects for hidden surface elimination;
4. Computation of surface normal vectors for surface shading calculation. A surface normal vector with the component in the direction of the screen equal to zero indicates that the surface has a silhouette edge.

In the following sections, each of the three surface representations will be considered.

3.1.1 Algebraic Representation

An algebraically defined surface is the set of points which satisfy the surface equation

$$f(x,y,z) = 0.$$

For simplicity, the homogeneous coordinate representation (wx, wy, wz, w) for the three dimensional points (x,y,z) will be used.

3.1.1.1 Surface Definition

A first order polynomial of the form

$$ax + by + cz + dw = 0$$

may be called a plane surface.

The defining equation may also be written in matrix form using homogeneous coordinates,

$$\underline{X}'\underline{A} = [x,y,z,w] \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = 0$$

where $\underline{X}' = [x,y,z,w]$ and $\underline{A}' = [a,b,c,d]$.

All points which satisfy the equation $\underline{X}'\underline{A} = 0$ lie on the plane. Furthermore, the plane divides the space into two halves determined by $\underline{X}'A < 0$ and $\underline{X}'\underline{A} > 0$.

A plane may be determined by three non-colinear points of the plane. The defining vector $\underline{A}$ may be determined from a solution of the equation

$$\begin{bmatrix} x_1 & y_1 & z_1 & w_1 \\ x_2 & y_2 & z_2 & w_2 \\ x_3 & y_3 & z_3 & w_3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = 0$$

where (x_1,y_1,z_1,w_1), (x_2,y_2,z_2,w_2), and (x_3,y_3,z_3,w_3) are the three points on the plane.

Since the above matrix equation is homogeneous, there are many solutions, all scalar multiples of each other. A convenient solution comes from the four, three by three subdeterminants of the matrix, developed by the application of Cramer's Rule and ignoring the scale factor

$$a = \begin{vmatrix} y_1 & z_1 & w_1 \\ y_2 & z_2 & w_2 \\ y_3 & z_3 & w_3 \end{vmatrix} , \quad b = -\begin{vmatrix} x_1 & z_1 & w_1 \\ x_2 & z_2 & w_2 \\ x_3 & z_3 & w_3 \end{vmatrix}$$

$$c = \begin{vmatrix} x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \\ x_3 & y_3 & w_3 \end{vmatrix}, \quad d = -\begin{vmatrix} x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \\ x_3 & y_3 & w_3 \end{vmatrix}$$

where $|\underline{M}|$ indicates the determinant of $\underline{M}$.

Another method of considering the three point solution to the plane is to form the 4 by 4 set of equations

$$\begin{bmatrix} x & y & z & w \\ x_1 & y_1 & z_1 & w_1 \\ x_2 & y_2 & z_2 & w_2 \\ x_3 & y_3 & z_3 & w_3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = 0$$

Since this set of equations is homogeneous, its determinant must be zero, i.e.

$$\begin{bmatrix} x & y & z & w \\ x_1 & y_1 & z_1 & w_1 \\ x_2 & y_2 & z_2 & w_2 \\ x_3 & y_3 & z_3 & w_3 \end{bmatrix} = 0$$

Expanding this determinant about the top row gives

$$\begin{bmatrix} y_1 & z_1 & w_1 \\ y_2 & z_2 & w_2 \\ y_3 & z_3 & w_3 \end{bmatrix} x - \begin{bmatrix} x_1 & z_1 & w_1 \\ x_2 & z_2 & w_2 \\ x_3 & z_3 & w_3 \end{bmatrix} y + \begin{bmatrix} x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \\ x_3 & y_3 & w_3 \end{bmatrix} z - \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} w = 0$$

Comparing this equation with $ax + by + cz + dw = 0$ gives the solutions.

If the solution yields a = b = c = d = 0, a degenerate plane, it indicates that the three points were linearly dependent, colinear.

An interesting computation is to determine the separating plant between two points, $(x_1, y_1, z_1, 1)$ and $(x_2, y_x, z_2, 1)$, which is the perpendicular bisector of the line between the points. The bisection point is

$$\left[\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}, \frac{z_1 + z_2}{2}, 1\right] .$$

The direction vector of the plane is

$$(x_1 - x_2, y_1 - y_2, z_1 - z_2, d).$$

The defining equation of the plane is therefore determined by:

$$\left[\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}, \frac{z_1 + z_2}{2}, 1\right] \begin{bmatrix} x_1 - x_2 \\ y_1 - y_2 \\ z_1 - z_2 \\ d \end{bmatrix} = 0$$

which leads to

$$d = -\tfrac{1}{2}[(x_1^2 - x_2^2) + (y_1^2 - y_2^2) + (z_1^2 - z_2^2)].$$

The separating plane is therefore defined by

$$[x, y, z, w] \begin{bmatrix} x_1 - x_2 \\ y_1 - y_2 \\ z_1 - z_2 \\ d \end{bmatrix} = 0.$$

Second order polynomials are called quadric surfaces and are defined by expressions in which each term has coordinates to the second power:

$$\begin{aligned} 0 = Ax^2 &+ 2Bxy + 2Cxz + 2Dxw \\ &+ Ey^2 + 2Fyz + 2Gyw \\ &+ Hz^2 + 2Izw + Jw^2 \end{aligned}$$

The quadric surface may also be described in matrix form:

$$(x\ y\ z\ w) \begin{bmatrix} A & B & C & D \\ B & E & F & G \\ C & F & H & I \\ D & G & I & J \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = 0$$

or

$$\underline{X}' \ \underline{Q} \ \underline{X} = 0 .$$

The shape of the surface depends upon the properties of the matrix $\underline{Q}$. Blinn lists the shape characteristics which may be determined from the signs of the eigenvalues of $\underline{Q}$. Since $\underline{Q}$ is a symmetric matrix, its four eigenvalues will be real valued; however, one or more may be zero. The sign sequences and shapes are given in Table 1. These interesting classifications are invariant to perspective transformations. Thus, although the ellipsoid, hyperboloid of 2 sheets and paraboloid are in the same category, they could always be distinguished from a cylinder or cone using the eigenvalue signs.

NUMBER OF NON-ZERO EIGENVALUES	EIGENVALUE SIGNS	SHAPE
4	+ + + + - - - -	No points satisfy the equation
	+ + + - - - - +	Ellipsoid, hyperboloid of 2 sheets, paraboloid.
	+ + - -	Hyperboloid of 1 sheet, hyperbolic paraboloid
3	+ + + 0 - - - 0	Only one point satisfies the equation
	+ + - 0 - - + 0	Cylinder, cone
2	+ + 0 0 - - 0 0	Single line
	+ - 0 0	Two intersecting planes
1	+ 0 0 0 - 0 0 0	Two coincident planes

Table 1. Eigenvalue shapes and signs.

The coefficient matrix $\underline{Q}$ for a given surface may be constructed from nine distinct points by solving the homogeneous equation:

$$\begin{bmatrix} x_1^2 & 2x_1y_1 & 2x_1z_1 & 2x_1w_1 & y_1^2 & 2y_1z_1 & 2y_1w_1 & z_1^2 & 2z_1w_1 & w_1^2 \\ & & & & \cdot & & & & & \\ & & & & \cdot & & & & & \\ & & & & \cdot & & & & & \\ x_9^2 & 2x_9y_9 & 2x_9z_9 & 2x_9w_9 & y_9^2 & 2y_9z_9 & 2y_9w_9 & z_9^2 & 2z_9w_9 & w_9^2 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \end{bmatrix} = \underline{0}.$$

3.1.1.2 Transformations

It is well known that a 4 by 4 homogeneous transformation matrix $\underline{T}$ may be used to perform translations, rotations, scaling, perspective and projective on a set of object points. If the object is represented algebraically by planar or quadric surfaces, then the entire object may be transformed by a matrix operation on its coefficient matrix.

For a planar surface defined by its normal vector $\underline{A}$, the transformed vector $\underline{A}'$ is obtained simply by

$$\underline{A}' = \underline{T}^{-1}\underline{A}$$

where $\underline{T}^{-1}$ is the inverse of the point transformation $\underline{T}$. Since the homogeneous representation contains an arbitrary scale factor, the adjoint matrix of $\underline{T}$ may be used in place of the inverse.

For a quadric surface defined by the matrix $\underline{Q}$, the transformed coefficient matrix $\underline{Q}'$ may be obtained as:

$$\underline{Q}' = \underline{T}^{-1^t}\,\underline{QT}^{-1} .$$

As an example, consider the general origin centered sphere described by

$$x^2 + y^2 + z^2 - r^2 = 0.$$

The corresponding matrix $\underline{Q}$ is:

$$\underline{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -r^2 \end{bmatrix} .$$

Suppose we wish to translate the sphere so that its center is at point (h, k, 0). The point transformation matrix $\underline{T}$ is:

$$\underline{T} = \begin{bmatrix} 1 & 0 & 0 & -h \\ 0 & 1 & 0 & -k \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} .$$

The inverse of this matrix is:

$$\underline{T}^{-1} = \begin{bmatrix} 1 & 0 & 0 & h \\ 0 & 1 & 0 & k \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformed surface matrix $\underline{Q}'$ may be computed from:

$$\underline{Q}' = \underline{T}^{-1^t} \underline{Q} \, \underline{T}^{-1}$$

$$\underline{Q}' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ h & k & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -r^2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & h \\ 0 & 1 & 0 & k \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\underline{Q}' = \begin{bmatrix} 1 & 0 & 0 & h \\ 0 & 1 & 0 & k \\ 0 & 0 & 1 & 0 \\ h & k & 0 & h^2+k^2-r^2 \end{bmatrix}$$

This matrix $\underline{Q}'$ may be interpreted in the algebraic form

$$(x - h)^2 + (y - k)^2 + z^2 - r^2 = 0$$

which is the equation of a sphere centered at (h, k, 0).

3.1.1.2 Intersections and Clipping

The intersection of two planes is a line in three dimensions. A simple way to demonstrate this is to solve the two plane equations. Let $\underline{A}_1 = (a_{11}, a_{12}, a_{13}, a_{14})'$ and $\underline{A}_2 = (a_{21}, a_{22}, a_{23}, a_{24})'$ represent the normal vectors of the planes. Then a point (x, y, z, 1) on the intersection line must simultaneously satisfy the equations:

$$a_{11}x + a_{12}y + a_{13}z + a_{14} = 0$$

$$a_{21}x + a_{22}y + a_{23}z + a_{24} = 0 .$$

Eliminating the variable z, equating and rearranging gives:

$$\left[\frac{a_{11}}{a_{13}} - \frac{a_{21}}{a_{23}}\right] x + \left[\frac{a_{12}}{a_{13}} - \frac{a_{22}}{a_{23}}\right] y + \frac{a_{14}}{a_{13}} - \frac{a_{24}}{a_{23}} = 0$$

This line is the silhouette of the three dimensional line. Points on the three dimensional line may be determined by solving for z in the original equations.

It is often useful to determine the intersection of a plane and a quadric surface. The resulting space curve or trace may be used to describe the surface characteristics. Three cases must be considered, since the plane and quadric surface may not intersect, may intersect only in a point, or may provide a trace. The case may be determined by evaluating:

$$[a, b, c, d]\underline{Q}^{-1} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

This evaluation is analogous to determining whether a point lies on the quadric surface. If the expression is zero, the plane is tangent to the surface. If positive, the plane intersects the surface. If negative, the plane does not intersect.

3.1.2 Surface Normals

The normal vector, $\underline{n}$, to the surface, $f(x,y,z,w) = 0$, is formed by taking partial derivatives, i.e.:

$$\underline{n} = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right]$$

and evaluating these at a point on the surface.

For the planar surface, the normal vector is simply:

$$\underline{n} = (a, b, c)' .$$

If C = 0, the plane has a silhouette edge. For the quadric surface, the normal vector may be computed by:

$$\underline{n}' = 2(x, y, z, w)\, \underline{Q} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

A silhouette edge is formed if the z component is zero.

Since most quadric surfaces are infinite in extent, the description of natural or man-made objects with pure algebraic expressions is somewhat impractical. However, it is interesting to note the predominance of planar and quadric surface segments in most man-made objects.

3.2 Point Set Representation

The most common form of surface representation is by a set of planar or polygonal facets. This representation can only approximate a curved surface and has only first order or surface continuity. Surface normals can appear quite discontinuous. The merit of the approach is simplicity both in surface definition and in the omission of surface normal information.

An arbitrary surface can be approximated by planar facets. One interesting example is the ancient Chinese burial mask by which an entire body was covered with about 2000, one inch square jade chips. A modern example was recently described by NASA, in which the heat shielding for the space shuttle in which about 10,000 chips of high temperature resistant material were used for the outer covering.

The basic mathematical device in this representation is a polygon in three dimensions. Each polygon is described by the coordinates of the vertices around its perimeter and the connection information for the set of vertices.

The transformation of an object represented by planar facets is very simple. One simply uses the homogeneous coordinate matrix to transform the list of vertices. The connection information does not change.

The clipping process involves determining if an object intersects a clipping plane in three dimensions. This involves testing each edge of the object against the plane and creating new vertices for the intersecting surfaces. To determine whether an edge between two vertices,

$$\underline{v}_1 = (x_1, y_1, z_1, w_1)' \quad \text{and} \quad \underline{v}_2 = (x_2, y_2, z_2, w_2)'$$

intersects the plane with normal vector $\underline{A} = (a, b, c, d)'$, one needs to determine if the vertices are on different sides of the plane. This may be done by computing:

$$\begin{bmatrix} x_1 & y_1 & z_1 & w_1 \\ x_2 & y_2 & z_2 & w_2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} .$$

If the signs of f_1 and f_2 are different, the edge intersects the plane. The values of f_1 and f_2 are proportional to the distance from the vertices to the plane. The proportional distance of the plane from vertex $\underline{v}_1$ to vertex $\underline{v}_2$ is given by:

$$\alpha = \frac{f_1}{f_1 - f_2} .$$

The actual coordinates of the intersection may be found from

$$(x, y, z, y)' = \alpha \underline{v}_1 + (1-\alpha)\underline{v}_2$$

The calculation of intersection points may change the connectivity of the vertices and may require creating new "capping faces" at the intersections. Sutherland [2] describes the clipping process and efficient algorithms for clipping. Baumgart [3] considers the general problem of intersecting one polyhedral object with another.

3.3 Parametric Representations

The final method of curved surface representation is the parametric surface. This approach appears to be the modern technique for generating realistic curved surfaces.

The parametric technique represents each coordinate point by a set of bivariate functions:

$$x = X(u,v)$$
$$y = Y(u,v)$$
$$z = Z(u,v)$$
$$w = W(u,v)$$

As the parameters (u,v) vary, the functions sweep out all points on the surface. Often, only a segment of the surface is used as a modelling element. For example, a patch segment formed by restricting the parameters to the range [0,1] may be used. The functions may in general be of any form; however, consideration will be restricted to simple bivariate, cubic polynomials.

Let us first consider a univariate function x(t) of the form of a cubic polynomial where the parameter t is restricted to the range [0,1]. This function may be written as

$$x(t) = x_3 t^3 + x_2 t^2 + x_1 t + x_0 \quad .$$

In vector form

$$x = [t^3 \;\; t^2 \;\; t \;\; 1] \begin{bmatrix} x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix}$$

The column vector of coefficients define the curve and must be specified to evaluate the function. However, it is difficult to geometrically interpret the values of the coefficients. One method for specifying the coefficients which has geometric appeal is to specify the position of the curve at equi-spaced values of the parameter t, 0, 1/2, 2/3, 1 as shown in Figure 3.1.

The function values at the selected points may be represented as:

$$\begin{bmatrix} x(0) \\ x(1/3) \\ x(2/3) \\ x(1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1/27 & 1/9 & 1/3 & 1 \\ 8/27 & 4/9 & 2/3 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix}$$

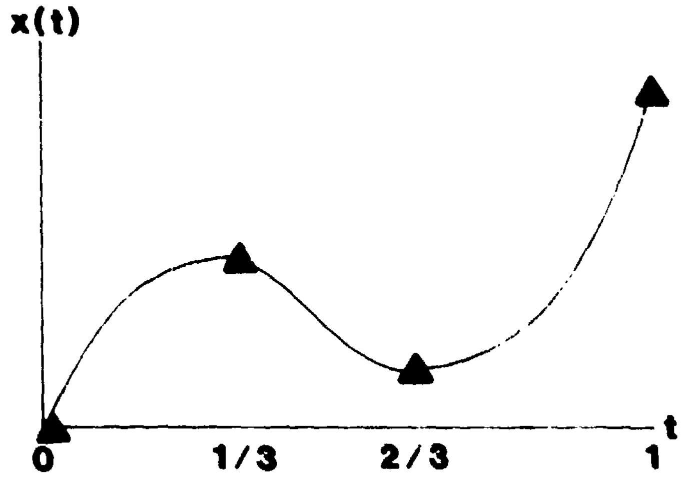


Figure 3.1. Specification of coefficient values for a cubic polynomial.

The required coefficients may be determined by inverting this matrix to obtain:

$$\begin{bmatrix} x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix} = \begin{bmatrix} -4.5 & 13.5 & -13.5 & 4.5 \\ 9.0 & -22.5 & 18.0 & -4.5 \\ -5.5 & 9.0 & -4.5 & 1.0 \\ 1.0 & 0.0 & 0.0 & 0.0 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1/3) \\ x(2/3) \\ x(1) \end{bmatrix}$$

The surface description for a bivariate cubic polynomial may be developed by first considering the x component in the representation:

$$x = [u^3 \; u^2 \; u \; 1] \; \underline{C} \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

where the 4 by 4 matrix $\underline{C}$ contains the 16 coefficients required for all cross terms in the bivariate cubic polynomial.

One method of specifying the $\underline{C}$ matrix is to select 16 control points at parameter values u,v = 0, 1/3, 2/3, 1. The matrix $\underline{X} = \{x(u,v)\}$ may be used to determine $\underline{C}$ by:

$$\underline{C} = \underline{M}\,\underline{X}\,\underline{M}'$$

where $\underline{M}$ is the inverse of the u or v selected point matrix previously given. Several other methods may also be used which have more geometric appeal.

For a curve with N points, a practical approach is to divide the curve into segments and fit a cubic polynomial to each segment. This avoids the oscillations often obtained with high order polynomials. Note that if a cubic polynomial was determined for each 4 point segment, the total curve would be continuous; however, the derivative would probably not be continuous at the boundaries of the segments. This slope discontinuity may be avoided by defining each cubic section in terms of two function values and two slope values, as shown in Figure 3.2. Since

$$x(u) = x_3 u^3 + x_2 u^2 + x_1 u + x_0$$

and

$$x'(u) = 3x_3 u^2 + 2x_2 u + x_1$$

at the control points u = 0,1 one may always develop four equations:

$$\begin{aligned} x(0) &= x_0 \\ x(1) &= x_3 + x_2 + x_1 + x_0 \\ x'(0) &= x_1 \\ x'(1) &= 3x_3 + 2x_2 + x_1 \end{aligned}$$

The polynomial coefficients may be determined from the solution to these equations which in matrix form is:

$$\begin{bmatrix} x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x'(0) \\ x'(1) \end{bmatrix} .$$

This technique generates a curve called the Coons [4] curve which may be generalized to Coons' surface description. The derivative continuity over a set of N points is maintained by overlapping the start and end

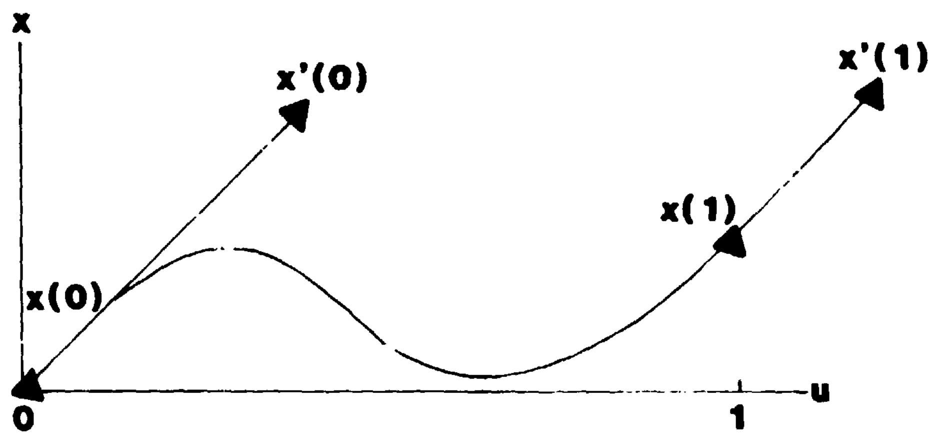


Figure 3.2. Specification of a curve by end point function and derivative values to generate a Coons curve.

points of the segments.

The generalization of the Coons approach to surface patches also requires the specification at each corner of the patch not only of the first derivatives in the u and v directions, but also a second derivative $d^2x/dudv$.

Another approach which achieves the derivative continuity but avoids the specification of the derivatives was developed by Bezier [5]. With this method, the cubic is again specified by four points of which two are the start and end points; however, the other two are points for which the end point tangents must pass through as shown in Figure 3.3.

The conversion between control point and coefficients is:

$$\begin{bmatrix} x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} .$$

The Bezier approach has several advantages. The shape of the approximating polygon gives a good intuitive approximation to the surface. Also, the curve always lies within the convex hull of the control points. The generalization to surface patches requires only a control net of 4 by 4 points. The surface patch will pass through the four corners of the net and stretch toward the other net points.

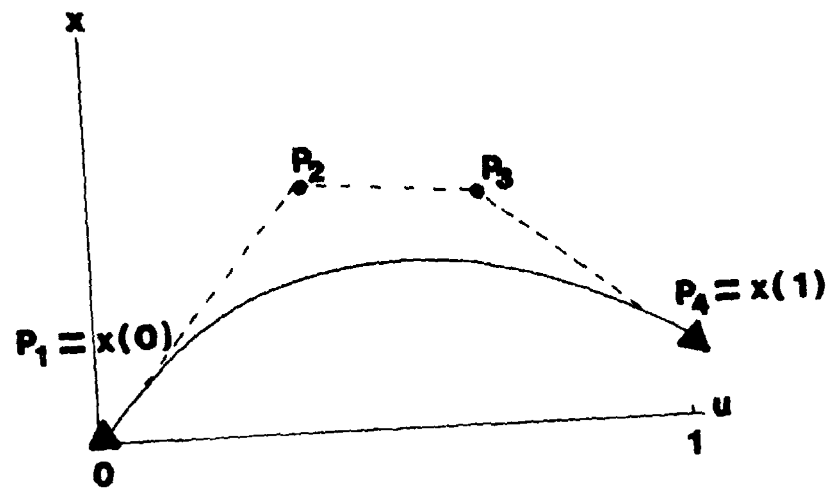


Figure 3. Specification of a Bezier curve by end points and bounding polygon.

Another surface representation method which allows second order continuity is the B-spline technique. This approach also requires four control points as shown in Figure 3.4, but generates a curve which may not pass through the control points. The matrix to convert control points to cubic polynomial coefficients is:

$$\begin{bmatrix} x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$$

The end points of the curve come close but do not necessarily pass through p_2 and p_3. The blending of curve segments is done by overlapping 3 control points on each segment. The surface description using B-splines uses a mesh of control points similar to the Bezier technique; however, the patch approximates the center square of the mesh instead of the outside boundaries.

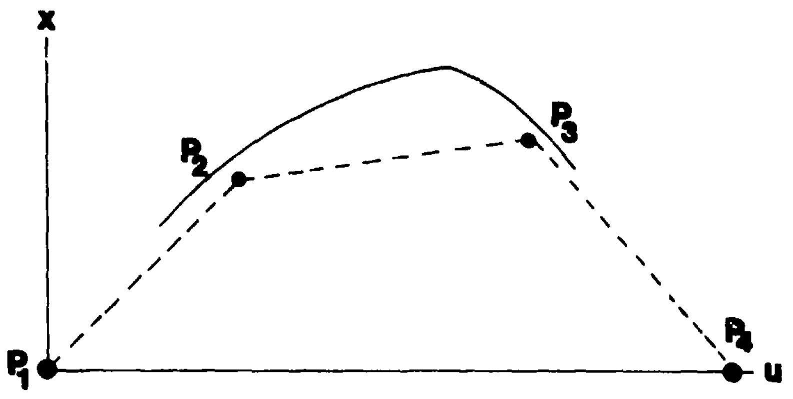


Figure 3.4. Cubic B-spline specification by control points.

3.4 Advanced Curved Surface Representation

Although a large amount of research has been done in the area of computer synthesis of surfaces, only a few hardware system are available that can produce "realistic" scenes at TV rates. One example is the excellent General Electric scene generation system. The fundamental difficulties encountered in realistic scene generation are that not only must the scene be efficiently represented in a computer data base, but it must also be efficiently manipulated to produce TV rate sequences of scenes. This involves solving the hidden surface problem, changing the resolution of the scene for a zooming sequence, high-speed transformations such as translations, rotations, and perspective, and shading or illumination calculations. Also, embedded in the solution are the problems of modelling arbitrary shaped surfaces and manipulating these models, for example, as in determining the intersection curve of two arbitrary shaped surfaces. Most previous solutions have been developed for polygonal surfaces or gridded data bases. Recent research in computation geometry for computer graphics indicates the possible feasibility of a free form surface representation using discrete B-splines.

The purpose of this section is to review this development to determine the application of this generalized representation to scene synthesis.

3.4.1 Parametric Surface Representation

Three general methods are available for surface representation: algebraic, point set, and parametric. An algebraic equation of the form $f(x,y,z) = 0$ is not considered appropriate for the free form surface. For example, no one could write a general equation to describe a mountain, a tree, or a grass lawn. The second form of representation, vertex point set, is most appropriate for polygonal facet representations which are implemented in state of the art systems and will not be considered in this review. The third method appears to offer the most promise for generating realistic curved surfaces, and the parametric approach will now be considered in some detail.

In the parametric techniques, each coordinate point is represented by a set of bivariate functions:

$$x = X(u,v)$$

$$y = Y(u,v)$$

$$z = Z(u,v)$$

$$w = W(u,v)$$

Only two parameters (u,v) are needed to sweep out all points on the surface since a surface in three dimensional space is by definition two dimensional (fractal surfaces excluded). The most common approaches restrict the parameters to a finite range such as $(0,1)$, so that the surface may be thought of as a patch, rather than a continuim. This leads to several questions. What equations should be used to describe the patch? How are the boundaries and coefficients of the patch specified? How does one insure continuity between patches? If one has a scene consisting of objects described by patches, is there an easy way to determine which object obscures the other? If only a partial obstruction occurs, how does one determine the intersection surface so that only the visible portion can be displayed? Is there an efficient technique for extending an object description from one resolution to another? For example, if one were to attempt to simulate an orange or a radome, then when viewed from a distance, the object should appear as a smooth sphere, but when viewed at close range, it should appear very textured. If algorithms have been developed which provide answers to these questions, then it may be possible to build a free form display system.

The problem just described may be called the major dilemma of scene synthesis. In order to accurately model free form surfaces, a data base of smooth functions, such as splines, are required. In order to produce a high speed, high quality, shaded, hidden surface display, a polygonal data base is required. It appears that one can have either a good model or good display, but not both. It should be noted that either solution has important applications. For example, in designing free form surfaces for computer aided manufacture (CAM) for control of a numerical milling machine, it may be sufficient to have a good model. Also, in many scene synthesis applications, the polygonal surface display is adequate. However, a scene synthesis system which could provide both a good model and a good display would certainly be desirable in both applications. A technique which would interface the two tasks efficiently by permitting a free form model to be approximated by a polygonal representation may be a reasonable solution, since this would permit a general data base to be shared between the model and display functions.

One important technique which should be incorporated in such a system has been called a subdivision, coarse to fine, or hierarchical technique. This technique would permit the designer to specify an initial set of control points, then proceed to add more control points in regions in which more detailed specifications are required without altering the other regions. For example, one might represent a quadratic function by a degenerate cubic function, then permit generalization to the cubic.

In a recent paper by Cohen, Lyche, and Riesenfeld [6], the mathematical theory of discrete B-splines and the Oslo subdivision algorithm are described. A summary of this paper will now be presented in an attempt to determine if this theory is sufficient to answer the previous questions.

3.4.2 B-Splines

The classical method for fitting a set of (n+1) data points might be to form a Lagrangean interpolating polynomial of degree n which passes through these points. However, if n is large, undesirable oscillation may occur in the polynomial which may have as many as (n-1) maxima and minima. The modern method of avoiding this problem is to construct a composite curve by fitting successively low-degree polynomials to successive groups of data points. The resulting piecewise polynomials will be continuous but may be discontinuities in slope at the joints between successive curve sigments. In most applications, this slope discontinuity will be unacceptable. By the use of polynomial splines, it is possible to interpolate the data smoothly using low degree polynomails to reduce oscillation problems.

For a set of (n+1) data points (x_i, y_i), i=0, 1, . . . , n, we wish to fit these points with a composite function N(x) such that:

i) over each subinterval M(x) is a cubic polynomial;

ii) M(x) and its first and second derivatives are continuous at the data points.

The resulting smooth piecewise cubic curve is called a cubic spline. Splines of higher degree result when continuity of the third or higher derivatives are specified.

To generate the cubic spline, we may start with the first pair of data points or span. The points x_i are often called knots. On the first span we have:

$$M(x_0) = y_0$$
$$M(x_1) = y_1$$

and

$$M(x) = ax^3 + bx^2 + cx + d.$$

In order to determine the four coefficients (a,b,c,d), two additional conditions are needed. Suppose that the first and second derivatives are known at x_0, i.e.:

$$M'(x_0) = y_0'$$
$$M''(x_0) = y_0''$$

Then since

$$M'(x_0) = 3ax_0^2 + 2bx_0 + c$$
$$M''(x_0) = 6ax_0 + 2b \quad ,$$

a set of simultaneous equations may be used to solve for the coefficients, i.e.:

$$y_0 = ax_0^3 + bx_0^2 + cx_0 + d$$

$$y_1 = ax_1^3 + bx_1^2 + cx_1 + d$$

$$y_0' = 3ax_0^2 + 2bx_0 + c$$

$$y_0'' = 6ax_0 + 2b \ .$$

Once (a,b,c,d) are determined, the polynomial may be used to generate $M'(x_1)$ and $M''(x_1)$. Thus, the four conditions necessary for determining the cubic coefficients for the next span can also be determined. The remaining segments of the spline may be successively calculated in this way until the last data point is reached. In practice, the spline may not be calculated in this way, because of roundoff error and difficulty in specifying the second derivative; however, a spline is easily constructed from the data values and two other conditions.

With these conditions in mind, consider the generation of a cubic spline which has the properties that $M(x) = M'(x) = M''(x)$ at each end. Over three spans, the solution comes out $M(x) = 0$. However, over four spans with a non-zero function value at an internal knot, the B-spline of order 4 or degree 3 results. The B-spline departs from zero only over precisely four spans. In general, the B-spline $M_{mi}(x)$ of order m (or degree m-1) on a given knot set is zero everywhere except over the m successive spans $x_{i-m} < x < x_i$. The B-spline is determined by the knot set and one additional value of y, which may be removed by normalizing the amplitude in some way. One such normalization gives the normalized B-spline:

$$N_{mi}(x) = (x_i - x_{i-m})\, M_{mi}(x) \quad .$$

The practical importance of B-splines is enhanced by the property that any spline of order m on the set of knots $x_0, x_1, \ldots, x_n$, can be expressed as a sum of multiples of B-splines defined on the same knot set extended (m-1) additional knots at each end. Changing the coefficients of one of a set of a B-spline curve alters precisely m spans of the curve. This permits local modifications which is a desirable feature in design.

3.4.3 Subdivision Methods

A method of describing a given curve through a set of knots in terms of another curve with a larger set of knots is called a subdivision technique. These techniques provide two fundamental advantages. First, they permit a designer to refine a design shape. Second, they provide a technique for obtaining a piecewise linear approximation to spline surfaces. After enough levels of recursive subdivision, the piecewise linear approximation to a smooth surface should be satisfactory. The linear approximation may then be used with existing shading, and hidden surface removal algorithms.

3.4.4 Discrete B-Splines

The discrete B-splines have values only at discrete points; however, when these points are connected, they resemble the profile of a B-spline basis function. They exhibit local support and are nearly symmetrical except at the ends of the support interval. If the discrete points are taken sufficiently dense, the B-spline results. The discrete B-splines are exactly the coefficients necessary to represent a coarser set of B-splines by a set over a refinement of a larger knot set. The Oslo algorithm is a recursive refinement formulation. The new points of a subdivision step are simply points along discrete spline defined by the old points.

3.4.5 Summary

The authors' approach to the free form scene synthesis problem provides:

1. a general curved surface representation;
2. a general hierarchical design strategy for subdividing; and
3. a method for developing a polygonal approximation for interfacing with existing display techniques.

Remaining questions regarding adaptive subdivision, termination criterion, efficient surface representation, and implementation are still open. However, the technique appears to be a promising solution to the dilemma of a good model and a good display.

3.5 Hidden Surface Algorithms

3.5.1 Background

To provide realistic synthesized images, the hidden surface problem must be solved. This problem of determining which surfaces of the objects in a scene are visible from a given viewing point has been said to be the most challenging problem in the field. The problem is not theoretically difficult and, in fact, Roberts' [7] algorithm provides one solution. However, this algorithm requires a large amount of computation time even for relatively simple scenes.

The requirements for an effective hidden surface algorithm are:

1. To accurately determine which surfaces of the objects in a scene are visible from a given viewing point. This also involves determining the visibility of intersecting surfaces of objects.
2. To eliminate hidden surfaces for both planar and curved object surfaces.
3. To operate at or near real time TV rates, in which a new image frame must be computed and displayed in 1/30 second, for both simple and complex scenes. This requirement may be satisfied by an algorithm which has a number of computations not exponentially related to the complexity of a scene, or by precomputation of object characteristics or by the use of frame-to-frame coherence.

Although the problem of hidden surface and hidden line elimination has been the subject of intensive research for almost 20 years, no clear cut optimum solution has been developed. However, at least two commercial systems, available from GE and Evans and Sutherland Companies, are available with hidden surface algorithm solutions.

The degree to which these available systems satisfy the above requirements is currently being investigated.

A review of several previous solutions to the hidden line and surface problem will now be presented to provide insight into the computational requirements and scene restrictions.

Robert's classic paper [7] included the first known solution to the hidden line problem for convex planar objects. Roberts introduced an elegant volume test to determine if each relevant edge was inside the volume occupied by some object in the scene. The test is implemented by first writing the parametric equation for a line from a point on the edge to the viewpoint:

$$\underline{x} = \alpha \underline{x}_0 + (1 - \alpha)\underline{x}_1$$

where $\underline{x}$ is a point on the ray, $\underline{x}_0$ is the point on the edge, $\underline{x}_1$ is the viewpoint, and α is the distance parameter, $0 \leq \alpha \leq 1$.

If the planar surface of an object is described by the vector $\underline{a}$ where

$$\underline{a}'\underline{x} = a_1x_1 + a_2x_2 + a_3x_3 + a_4 = 0$$

then the inner product $\underline{a}'\underline{x}$ may be tested to determine if the point $\underline{x}$ lies on the plane or on the negative or positive side of it, i.e.:

$$\underline{a}'\underline{x} = \begin{cases} < 0, \text{ negative side} \\ = 0, \text{ on plane} \\ > 0, \text{ positive side} \end{cases}$$

If the vectors $\underline{a}$ for each side of a planar solid are collected in a matrix, $\underline{M}$, and the normal vectors to each plane oriented to point outward, then the product $\underline{M}'\underline{x}$ provides a vector, $\underline{V}$.

If all components of $\underline{V}$ are negative for any value of α, then the point $\underline{x}$ is inside the object and thus hidden from view.

The total Roberts algorithm did not test each value of α, but used efficient techniques to determine bounding boxes for each object to eliminate comparisons of collections of objects with other objects. However, the computation required by the Roberts algorithm grows roughly as the square of the number of objects in the scene. Also, the scene must be decomposed into planar, convex objects.

Another important paper on a hidden line solution for the orthographic display of both planar and quadric surfaces was presented by Weiss [8]. Quadric surfaces of the form:

$$\begin{aligned} Q(x,y,z) = {} & a_1x^2 + a_2y^2 + a_cy^2 \\ & + b_1yz + b_2xz + b_3xy \\ & + c_1x + c_2y - c_3z + d = 0 \end{aligned}$$

were considered and methods were developed for determining bounding regions for the surfaces, intersections of surfaces, and visible edges from any viewpoint. A point-by-point calculation and test of each point for visibility against all the surfaces was made. This process was very time consuming. The inclusion of both planar and quadric surfaces permitted some very interesting images to be generated.

The key task in the hidden surface solution is the determination of the visibility or invisibility of the surface points. A simple surface visibility test for convex polyhedron objects is to determine if the angle between the local line of sight and the outward normal of a face is greater than 90^0. If this angle is greater than 90^0, the face may be declared invisible. Also, any line which is the intersection of two invisible faces may be declared invisible. This surface test cannot be extended to other types of solids, since the surfaces may be partially hidden on non-convex objects. The method presented by Appel [9] may be characterized as midway between the surface visibility test and the point visibility test. The fundamental idea of Apple's quantitative invisibility is that it is not sufficient to specify a curve invisible, but that the total number of visible surfaces that hide a point on the curve should be measured, and when no surface hides the points on the curve, the curve is considered visible. The quantitative invisibility of a point on a curve is the number of visible surfaces which hide the point. Visible points have a quantitative invisibility of 0. Apple's algorithm introduced an efficient idea called edge coherence. The quantitative invisibility of a relevant edge can change only where the projection of the edge crosses the projection of some surface contour edge. At this intersection, the quantitative invisibility increases or decreases by 1. Furthermore, if the quantitative invisibility of the initial vertex point of an edge is known, the visibility at any point can be calculated by summing the quantitative invisibility changes. A number of singularity situations, such as when the image of a vertex lies on the image of an edge, can occur and require special attention in computing the invisibility.

The previous algorithms may be called object space methods in that the visibility test is made in the three dimensional object space. The next class of algorithms to be considered may be called image space algorithms, since the visibility of points in the projected two dimensional image is determined. The image space algorithms are designed to create images for a fixed resolution digital display, such as a 512 by 512, digital TV system. The motivation for these studies is the production of realistic, real time TV images.

One of the first efforts in the development of an image space algorithm was begun by Schumacker and his collaborators at General Electric in 1965. This work led to the development of the first real time solution to the hidden surface problem and has been operational since 1968 (Sutherland, Sproull, and Schumacker [10], and Bunker and Heeschen [11]

An interesting development in image space algorithms is the concept of overwriting, which was used to display transparent objects by Newell, et. al. [12] . A priority list which is used to determine which face is visible when a number of faces surround a given pixel may also be used in a different way. If the images of successively higher priority faces are written in the image buffer, then faces of higher priority will overwrite faces of lower priority and a correcc hidden surface view will be produced. Newell achieved the transparency effect by permitting transparent faces to only partially overwrite the underlying face. If the intensity of the transparent face is greater than that of the underlying face, the transparent face intensity is used directly. Otherwise, a linear rule is used to combine the two intensities.

The concept of a list-priority or visibility ordering of all surfaces in a scene was introduced in the Schumacker work [13]. The priority of a surface can be expressed as a linear ordering of the surfaces such that if two surfaces need to be compared for visibility, the one with the higher priority is the visible one. The possibility that the priorities can be assigned relatively independently of the viewpoint, and that the scene can be divided into clusters corresponding to the faces of objects, make the list priority approach very efficient. Furthermore, much of the priority computation can be done as preprocessing, and would not change significantly from frame to frame.

The implementation of the list-priority algorithm requires the computation of face-priorities within each cluster, the cluster-priority computations, and the image generation.

The priority assignment may be described as follows. If, from any viewpoint, the back faces of an object with respect to that viewpoint are eliminated, then integer number face priorities may be assigned to each face in an object. A cluster is defined as a collection of faces that can be assigned a fixed set of priority numbers which, after the back edges are removed, provide correct priority from any viewpoint. The face priorities assignment requires computing whether face A can, from any viewpoint, hide face B. If so, then face A has priority over face B. The face computation must be made for all faces in each cluster. A priority graph is then constructed. If there are circuits in this graph, the faces cannot be assigned priorities and the cluster must be divided into smaller clusters.

The calculation of cluster priorities can be described in terms of a set of separating planes between the clusters. If the viewpoint lies in a region bounded by the piecewise linear separating plane, the cluster priority can be determined for each cluster by determining the visibility from that viewpoint.

The generation of the image may be accomplished with special purpose hardware which performs the following operations.

1) The cluster priorities are computed by comparing the viewpoint location to the separating planes in the scene.

2) A list of faces is constructed, in priority order, with back faces excluded.

3) The perspective coordinates of each edge in the scene are computed.

The final operations include a determination of which faces intersect a particular pixel, and a priority search to determine the visible face at this pixel.

A final class of hidden surface algorithms which will be considered are called scan line algorithms, as exemplified by the Watkins scan line method [14]. This is now commercially available from the Evans and Sutherland Computer Corporation. This algorithm is also implemented in the MOVIE.BYU software package. Sutherland describes the algorithm in terms of sorting. First a Y sort is performed to limit the consideration of the algorithm to a single scan line. The edges which project on the scan line are then sorted by their X coordinates to determine a set of edge spans which are then determined by edge crossings. The segments in this span are then sorted by their Z coordinates to determine which is visible.

In summary, a variety of solutions to the hidden surface problem have been developed and two have been implemented into hardware/software systems. The various algorithms exploit various forms of coherence in the scene to obtain efficient computations. Sutherland lists the possible forms of coherence as follows.

Frame coherence: The image does not change significantly from frame to frame.

Object coherence: Individual objects are confined to local volumes which may not conflict. The use of clusters is a form of object coherence.

Face coherence: The faces are generally small compared to the image size and may not conflict. Moreover, the penetration of faces is a relatively rare occurrence.

Edge coherence: The visibility of an edge changes only where it crosses another edge.

Implied edge coherence: If face penetration is detected, the location of the entire implied edge can be extrapolated from two penetration calculations.

Scan line coherence: The set of span segments encountered on one scan line and their X intercepts are closely related to those on the previous scan line.

Area coherence: A particular element in the output image and its neighbor elements are likely to be influenced by the same face.

Depth coherence: The different surfaces at a given screen location are generally well separated in depth relative to the depth range of each.

The previously described algorithms use various forms of coherence. Roberts' algorithm uses object coherence. The Apple algorithm relies heavily on edge coherence. The Schumacker algorithm makes use of depth coherence in the determination of separating planes and the establishment of a priority list. This algorithm also makes use of implied edge coherence by dividing polygons wherever they intersect. The Schumacker algorithm also makes effective use of frame coherence. Finally, the Watkins algorithm makes effective use of edge coherence.

3.5.2 Computational Considerations

The hidden surface problem is very costly, in terms of computation time. The effect of square law growth in the number of computations with the scene elements relegate the point by point methods to historical value only. The use of various types of coherence between frames, objects, edges, etc. seems to be a principal component in making an efficient algorithm. Also, the concept of dividing the computation into a pre-processing stage and an image production stage seems to be an important element in developing a real time algorithm. Sutherland stresses the idea of efficient sorting throughout the hidden surface algorithm and developed a classification of algorithms based upon the order in X, Y, and Z in which sorting was used in the algorithm. It appears that further improvements based upon the improved use of coherence are still possible in all the approaches.

3.6 Surface Shading

An important feature of an image synthesis system is surface gray scale prediction, or shading. Assuming that a hidden surface removal algorithm has been implemented, the next step toward realistic display involves the assignment to every picture element in the image, an intensity or color which accurately simulates the viewing situation. For simplicity, the intensity range will be assumed to be $0 \leq I \leq 1$ for monochrome shading. Furthermore, if the total intensity is due to several components, these will be assumed to add linearity to produce a total intensity.

The shading intensity depends upon the surface geometry. One important surface property for shading is the surface normal vector. If the surface is a plane represented in the form:

$$ax + by + ca + d = 0$$

a normal vector is simply (a, b, c). This concept can also be extended to curved surfaces by considering the normal vector to the tangent plane at a point. In polygonal surface representations especially at vertex points, a unique normal vector may not exist. In these cases, it may be convenient to consider an average normal vector.

Given a normal vector and the position of an illumination source, how does one determine the intensity of the corresponding point? The answer depends a great deal on the surface properties such as the spectral reflectance which determines how the surface reflects light of a specific color, the texture which determines if the surface is a diffuse or specular reflector, and the transparency of the surface which decreases the amount of light reflected.

One simple model for simulating an ideal diffuse reflector is based upon Lambert's Law[15] which states that a surface will diffuse incident light equally in all directions. The differences in intensity are produced by the different amounts of incident light per unit area intercepted by portions of the surface at various angles to the light source. The amount is proportional to the cosine of the angle between the normal vector, $\underline{N}$, and the vector to the light source, $\underline{L}$, as shown in Figure 3.5. The cosine may be computed as the inner or dot product of the two unit vectors. If the cosine is negative, it indicates that the viewer is on the opposite side of the surface from the light source and the intensity should be set to zero for an opaque surface. The intensity is given by:

$$I = \begin{cases} \cos^n\theta & , -\pi/2 < \theta < \pi/2 \\ 0 & , \text{otherwise} \end{cases}$$

where n = 1 for the normal Lambertian surface and n = 2 gives additional emphasis for small angles.

Specular reflectance or highlights may also be modeled. A simple model introduced by Phong[16] makes use of the fact that for any real surface, more light is reflected in a direction making an equal angle of incidence and reflectance. The additional light reflected in this direction is called the specular component. For example, if the surface were a perfect mirror, light would only reach the eye if the surface normal, $\underline{N}$, pointed halfway between the source direction, $\underline{L}$, and the eye direction, $\underline{E}$, as shown in Figure 3.6. This preferred direction has been called the direction of maximum highlights, $\underline{H}$, where

$$\underline{H} = \frac{\underline{L} + \underline{E}}{2}$$

For less than perfect reflectors, the specular component falls off slowly as the normal vector moves away from the maximum highlight direction. The Phong shading function is given by:

$$I = \begin{cases} \cos^m \phi & , \ -\pi/2 < \phi < \pi/2 \\ 0 & , \ \text{otherwise} \end{cases}$$

where ϕ is the angle between the normal vector $\underline{N}$ and the highlight direction $\underline{H}$, and m is a measure of the shininess of the surface with typical values between 50 and 60.

Another model of specular reflectance was developed by Torrance and Sparrow [17]. In this model, the surface is assumed to be composed of a collection of mirror-like micro facets oriented in random directions. The light reflected specularly can come only from the facets oriented to reflect in that direction. A Gaussian distribution was assumed for the average number, n, of facets oriented at an angle, x, from the average normal to the surface, i.e.

$$n = \exp\{-(\alpha/\sigma)^2\}$$

where σ is a property of the surface being modeled with large values

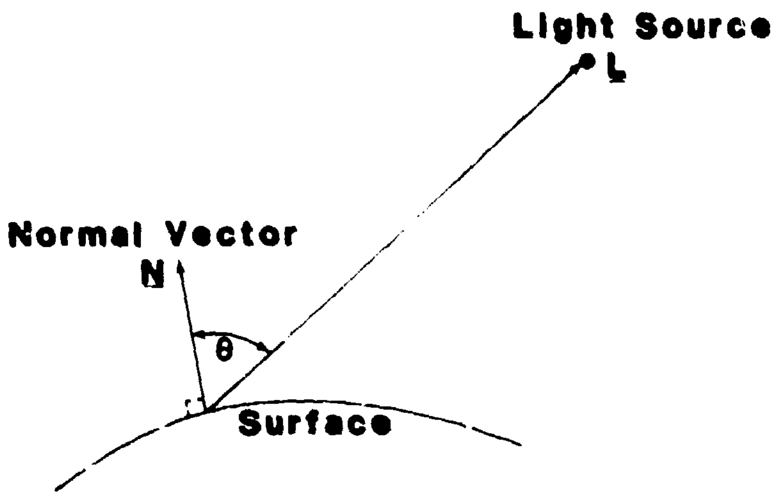


Figure 3.5. Lambert's Law for diffuse reflector.

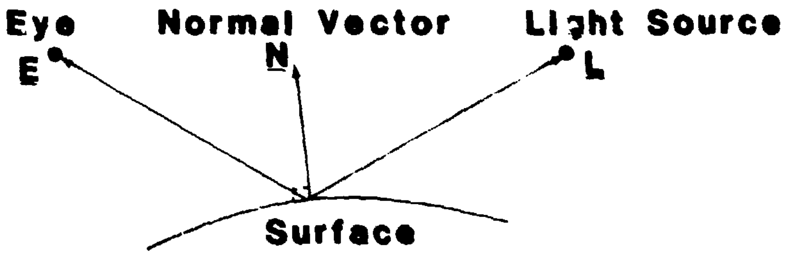


Figure 3.6. Specular reflectance.

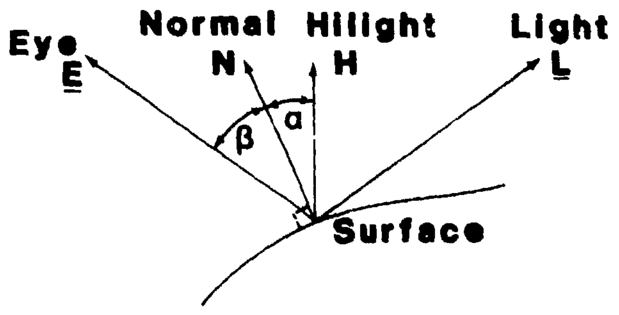


Figure 3.7. Torrance-Sparrow model.

yielding shiny surfaces and small values yielding dull surfaces. The angle α is the angle between the normal, $\underline{N}$, and the highlight direction, $\underline{H}$.

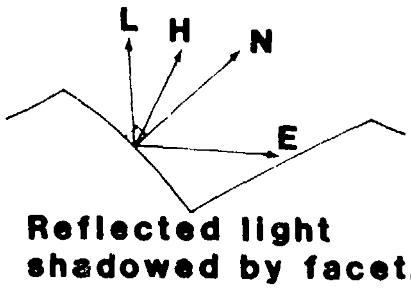
Another term is used to account for the fact that the observer sees more of the surface area when the surface is tilted. The increase in area is inversely proportional to the cosine of the angle of tilt, which is the angle between the average surface normal, $\underline{N}$, and the eye vector, $\underline{E}$. This is implemented by dividing by $\cos\beta$ where β is the tilt angle as shown in Figure 3.7.

Another effect which is included in the Torrence-Sparrow model is called the geometrical attenuation factor, G. The value of G which ranges between 0 and 1 represents the proportional amount of light after the masking of incident light or the shadowing of reflected light by the micro-facets as shown in Figure 3.8.

The final factor in the specular reflection is the Fresnel reflection which gives the fraction of light incident on a facet which is actually reflected rather than absorbed. The Fresnel coefficient, F, is a function of the angle of incidence on the micro-facet and the index of refraction of the surface is given by:

$$F = \tfrac{1}{2}\left[\frac{\sin^2(\phi-\theta)}{\sin(\phi+\theta)} + \frac{\tan^2(\phi-\theta)}{\tan(\phi+\theta)}\right]$$

where $\sin\theta = \sin\phi/n$, n = index of refraction, and ϕ is the angle of incidence or angle between $\underline{L}$ and $\underline{H}$ or $\underline{E}$ and $\underline{H}$.



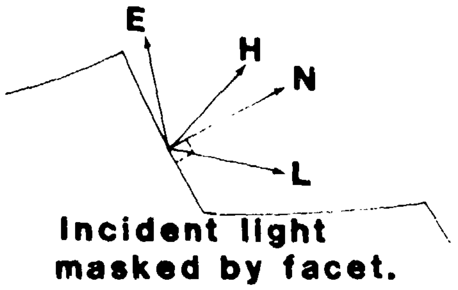
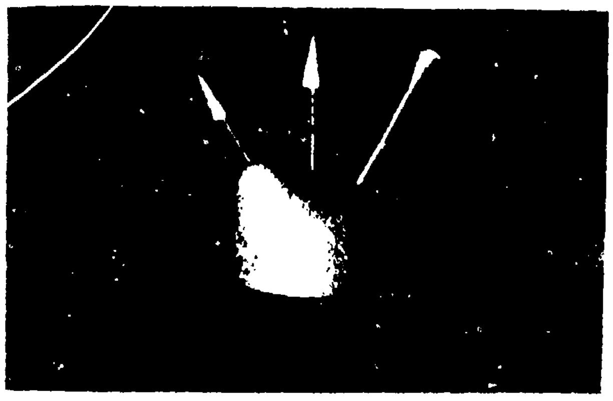


Figure 3.8. Masking and shadowing by micro-facets.



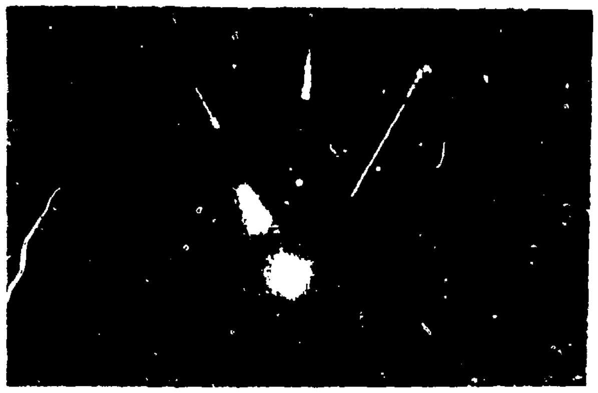


Figure 3.9. Comparison of Phong and Torrance-Sparrow reflection distributions for incident light at 30° from normal.

For metals which correspond to large values of n, the value of F is close to 1. For non-metals corresponding to small values, F is close to an expon-ntial function with values near zero at $\phi = 0$ and near one at $\phi = \pi/2$.

The total Torrence-Sparrow model for specular reflectance is given by:

$$I = \frac{nGF}{\cos\beta}$$

A comparison of the Phong model and the Torrance-Sparrow models is shown in Figure 3.9. There is a noticeable difference primarily for non-metallic and edge lit objects.

In addition to the surface characteristics inclu-ed in the previous models, several other features of the viewing situation may be important. One effect is due to the number of light sources. Even when a single light source is predominant, there may be a significant amount of ambient light, perhaps due to reflection from other surfaces. This effect may be modeled by dividing the received light, I_r, into two parts, one reflected from the surface, I_s, and the other due to ambient light, I_a. Thus, given

$$I_r = I_s + I_a$$

only the portion I_s would be modified by the surface characteristics.

Another effect is due to spectral reflectance. For visible light, three color components such as red, green, and blue or hue, brightness, and saturation must be computed as the product of the incident spectral energy and the surface spectral reflectance.

For example,

$$R = I_s \cdot r$$

$$G = I_s \cdot g$$

$$B = I_s \cdot b$$

where (r, g, b) are the tristimulus values of the surface at a given wavelength.

Another feature of color is that diffuse and specular reflectance are modified differently. Only the diffuse component is modified by the spectral reflectance. The specular highlights appear white.

The distance from the surface to the receiver is also important. The inverse square law of received light energy with distance accounts for the observation of two identical surfaces from different distances appearing different. The shaded surface appearance is not too sensitive to the particular form of inverse distance rule. For example, either the light source to surface distance may be used as R to make identical surfaces appear different. Also, various powers of R such as R^2 or R^4 have been used.

Fog attenuation also depends on the above distances and tends to compress the contrast range of the received light. Transparency also decreases the amount of light but not as a function of distance, but only as a function of the surface.

Finally, the difference between the various shading algorithms should be noted. A faceted shading algorithm is the simplest to implement since it assumes that the intensity is constant over an entire planar polygonal facet. This produces a shading discontinuity at polygon edges which is enhanced by the psychovisual Mach phenomena. Thus, this algorithm is not well suited for smooth surfaces. Another algorithm, called Gourard shading, is illustrated in Figure 3.10 and 3.11. The reflected intensity is computed at each polygon vertex using an average normal vector of the polygons meeting at the vertex. Then, for other surface intensities, a bilinear interpolation between the nearest vertices is computed. Although this algorithm is a significant improvement over faceted shading, the discontinuities in the derivative of the intensity produce some visible distortions.

Finally, the Phong algorithm will be considered. In this algorithm, the vertex normals are interpolated across a polygon surface. This minimizes the first order discontinuities and permits the computation of specular reflection. The total algorithm takes into account spectral reflection, incident and ambient light, specular reflectance, and smooth shading, and has been used to produce excellent simulated images.

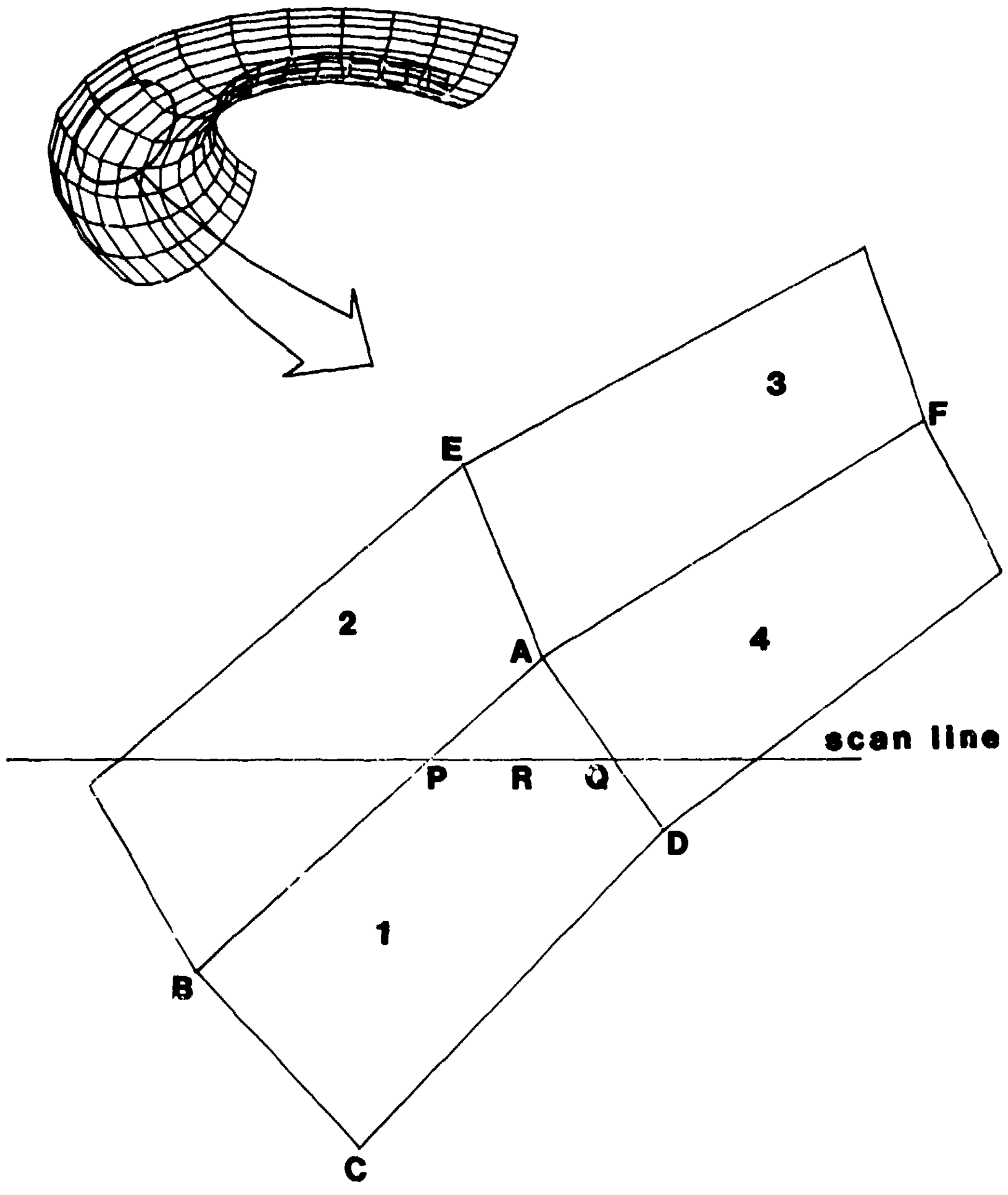
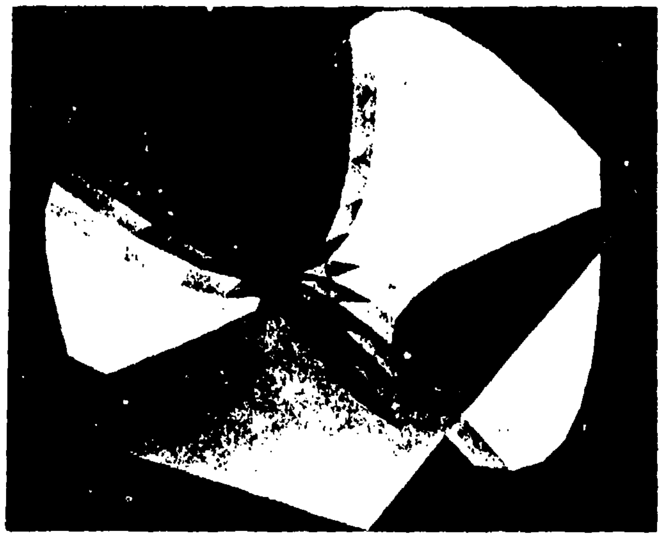
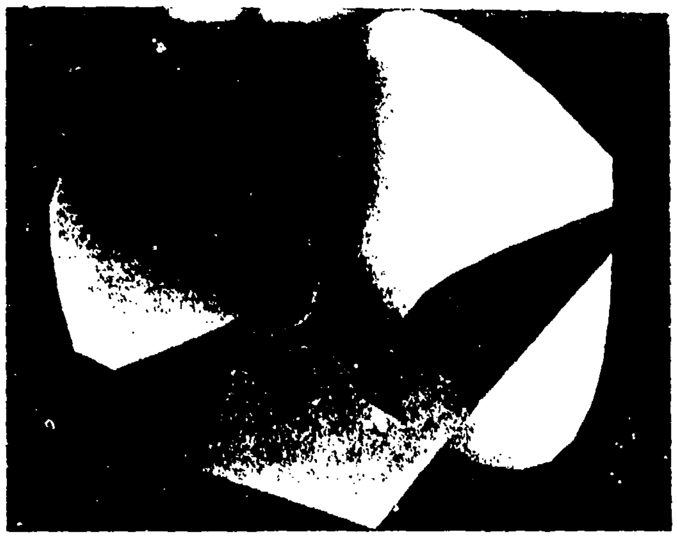


Figure 3.10. Gouraud or 'smooth' shading.



faceted shading



Gouraud shading

Figure 3.11. Two representations of the same surfaces.

3.7 Texture Synthesis

It is now possible to add realism to computer synthesized images by the addition of texture. One method for real time display of texture was recently reported by Schachter [18]. The mathematical model for the simulation of man made texture patterns such as urban building patterns, industrial areas and farmland is also considered by Ahuja and Schachter [19], and is a Gaussian random field called the "long crested wave model." With this model, a texture pattern $t(x,y)$ is formed from the sum of a background gray level b and a zero mean function $g(x,y)$. The pattern formed from a single long crested sinusoid is given by

$$t(x,y) = b + g(x,y)$$
$$t(x,y) = b + B\cos(ux+vy+\phi)$$

where B is the amplitude, u and v are the spatial frequencies and ϕ is a phase shift which is uniformly distributed in the interval $(0,2\pi)$. This single long crested wave might be used to model a plowed field. A pair of long crested sinusoids of equal amplitude produces a pattern whose structure does not depend on the phase difference between the waves. The pattern formed by the sum is described by

$$\begin{aligned} g(x,y) &= B\cos(u_1x+v_1y) + B\cos(u_2x+v_2y) \\ &= 2B\cos(u_ax+v_ay)\cos(u_bx+v_by) \end{aligned}$$

where $u_a = (u_1-u_2)/2 \qquad v_a = (v_1-v_2)/2$
$u_b = (u_1+u_2)/2 \qquad v_b = (v_1+v_2)/2$

The pattern formed by two long crested waves might appear as a rectangular pattern. When three or more waves are involved, the resulting pattern becomes an increasingly complex function of the amplitudes and frequencies of the components. The long crested wave model exemplifies the approach which may be called a frequency domain description of texture patterns.

The method of texture description based upon a Fourier analysis is illustrated in Figures 3.12 and 3.13, in which a set of texture patterns and the corresponding Fourier transform magnitudes are shown. If each texture pattern is represented by t(x,y), the corresponding Fourier transform magnitude |T(u,v)| is given by

$$T(u,v) = \iint_{-\infty}^{\infty} t(x,y)\exp\{-j(ux+vy)\}dxdy \quad .$$

The possible advantage of the Fourier description of texture is also illustrated in the transform patterns in Figure 3.12, such as 3.12(a) , the tapestry pattern. Note that in the spatial domain, the tapestry pattern is very difficult to describe. Horizontal linear segments are visible as well as a salt and pepper dot pattern. However, neither of these patterns exhibit enough regularity to permit a concise description. The only apparent spatial domain description would be a point by point specification of t(x,y). Now consider the corresponding Fourier transform pattern shown in Figure 3.13(a). This pattern has a definite rectangular grid structure with replica spectra at the grid intersections. Thus, the Fourier pattern appears to have a parameterization in terms of the distance between grid lines in the u and v directions, the replica spectra, and perhaps a random variation of these parameters.

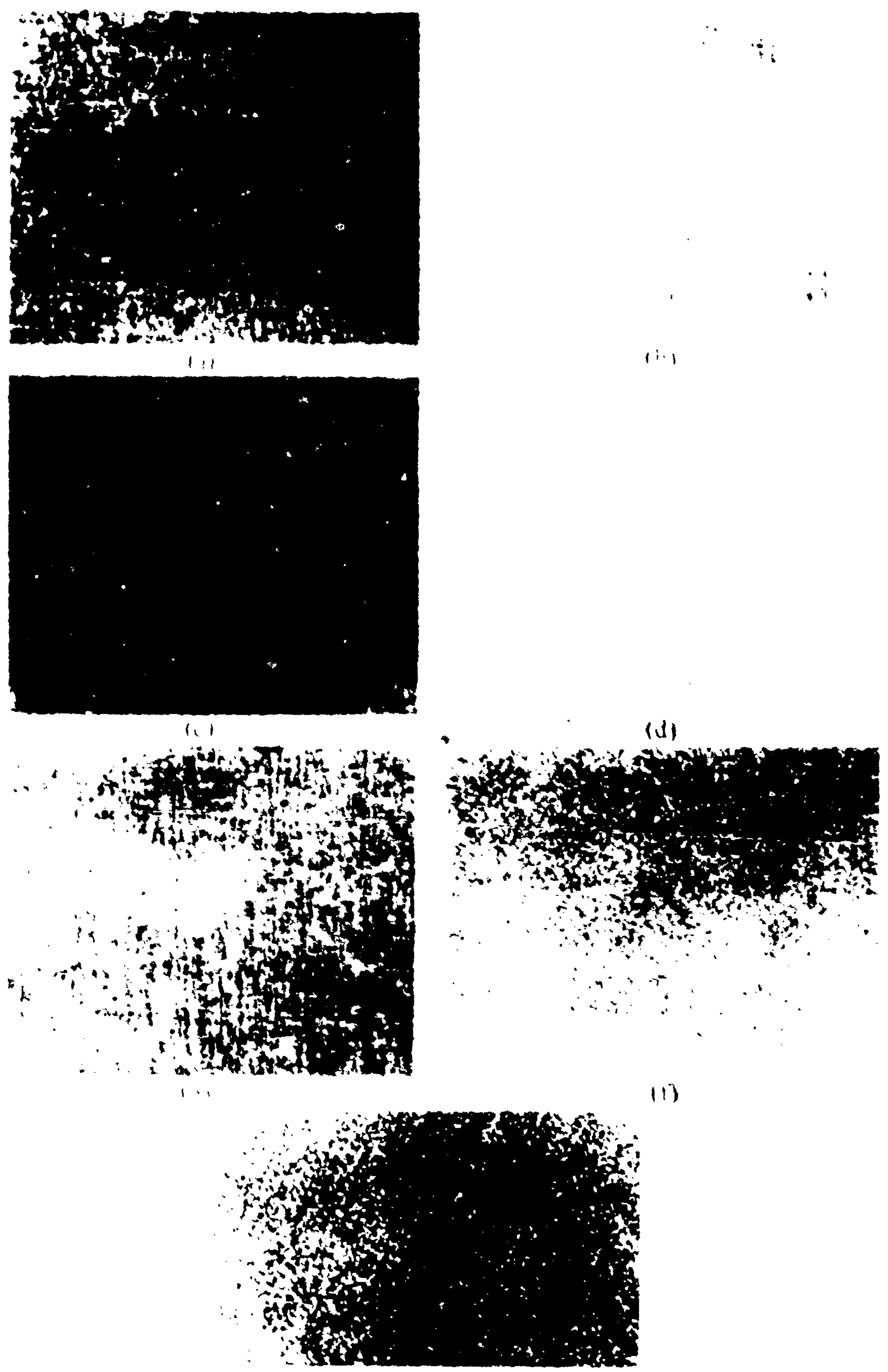


Figure 3.12. Spatial texture patterns.

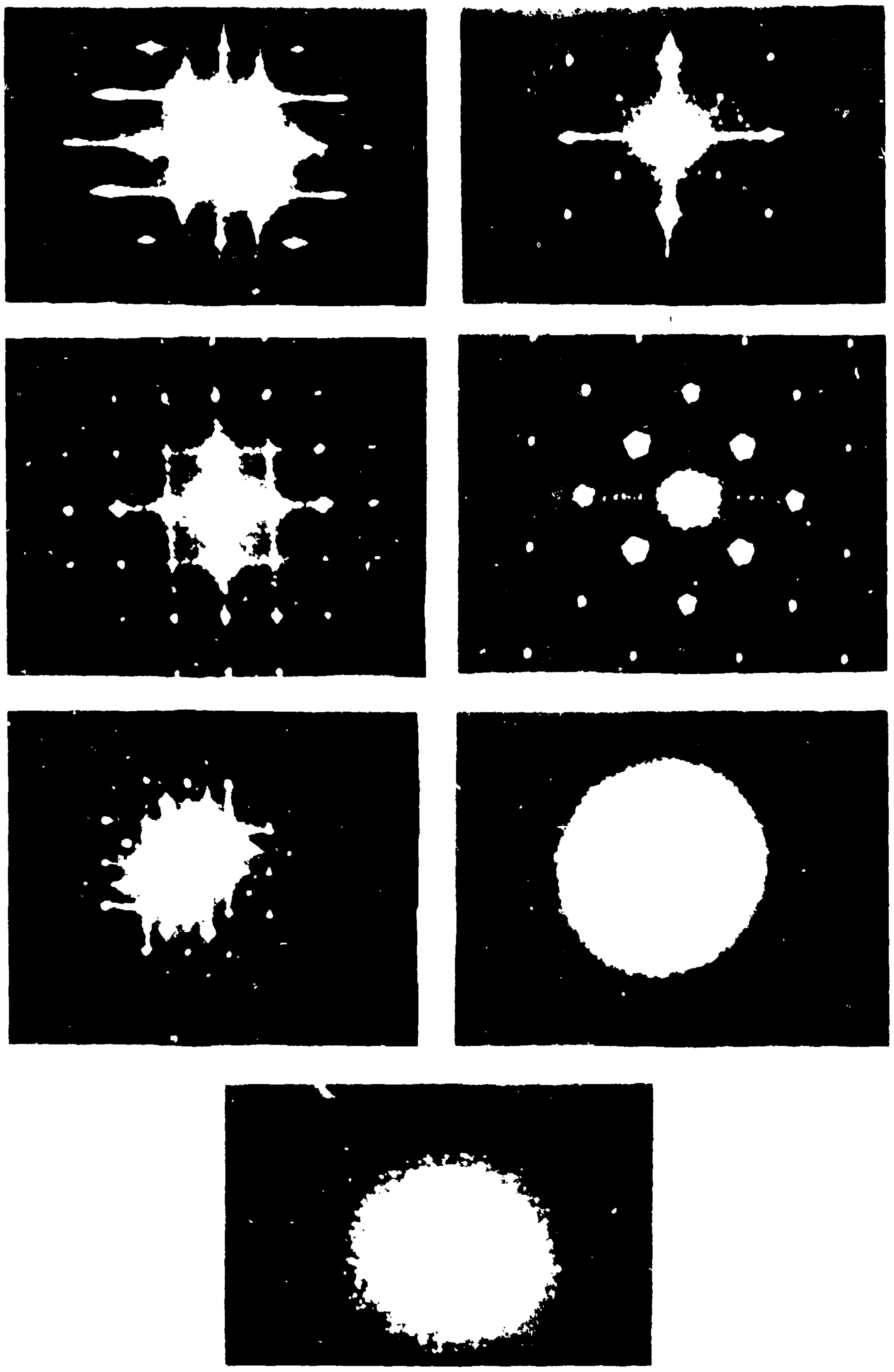


Figure 3.13. Fourier transforms of texture patterns.

Several similar illustrations may also be found in Harburn, Taylor and Welberry [20]. The patterns shown in Figure 3.14 are variations of square lattices. The perfect array shown in Figure 3.14(a) has a perfectly spaced transform with spectra spacing inversely proportional to the spatial dot spacing. In terms of Fourier theory, this spatial pattern may be considered as being composed of three functions. The basic circular dot function, c(x,y), shown in Figure 3.16(a), is convolved with an infinite impulse train of the form

$$i(x,y) = \sum_{-\infty}^{\infty} \sum_{-\infty}^{\infty} \delta(x-mX,y-nY)$$

which is shown in Figure 3.16(b). The result of this convolution is to copy the dot at the location of each impulse, i.e.

$$s(x,y) = \sum_{-\infty}^{\infty} \sum_{-\infty}^{\infty} c(x-mX,y-nY)$$

as shown in Figure 3.16(c). The final step is to multiply this result with an aperature function a(x,y) of the form

$$a(x,y) = \begin{cases} 1, & |x| \leq X_1 \text{ and } |y| \leq Y_1 \\ 0, & \text{otherwise} \end{cases}$$

where $X_1 > X$ and $Y_1 > Y$. The resulting dot pattern t(x,y) may be written as

$$t(x,y) = a(x,y)s(x,y) \quad .$$

The process may be more clearly understood in the Fourier transform domain. Denoting the Fourier transforms of c, i, s, a, and t as C, I, S, A, and T, it is clear that the transform pattern shown in Figure 3.15(a)

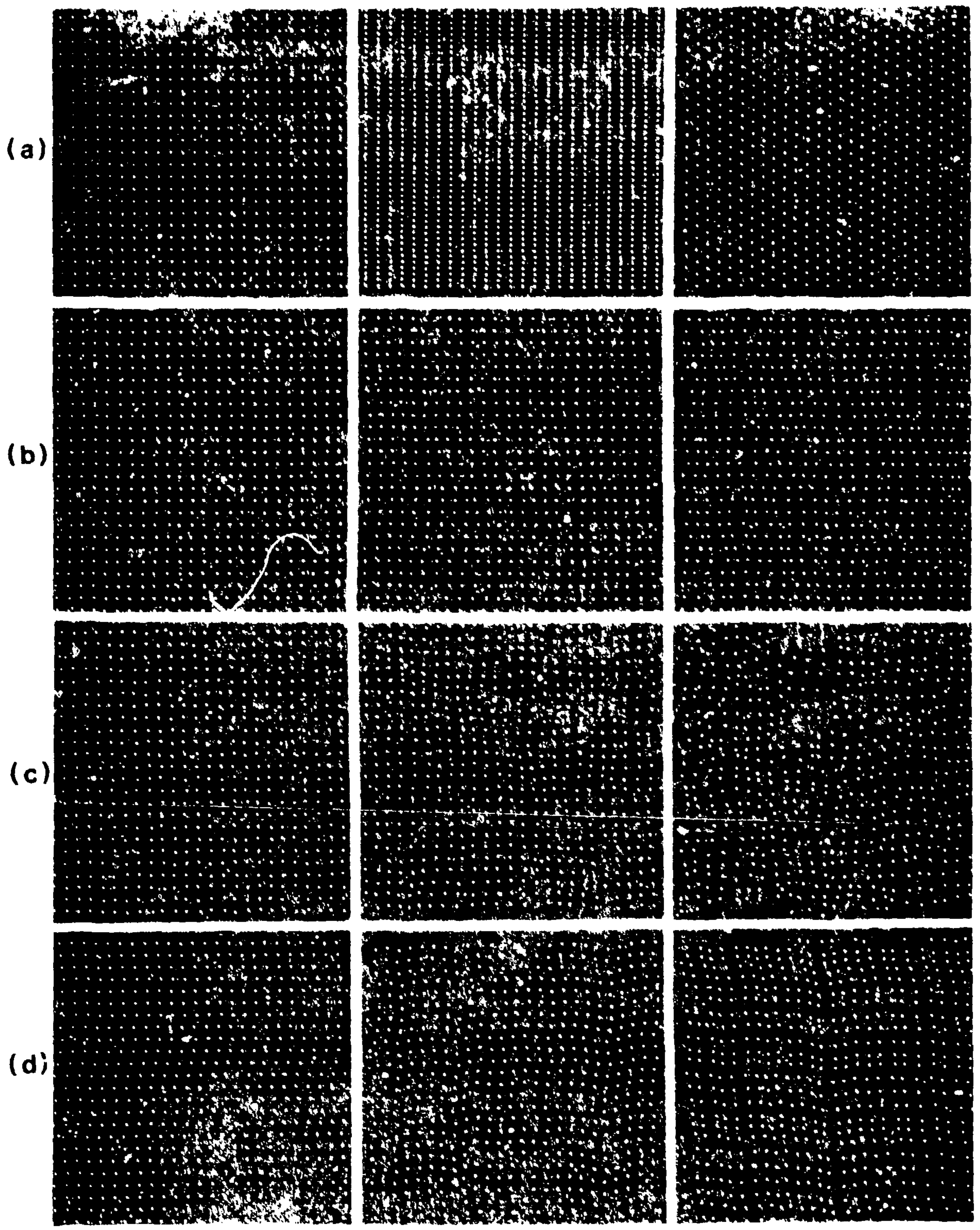


Figure 3.14 Spatial aliasing patterns

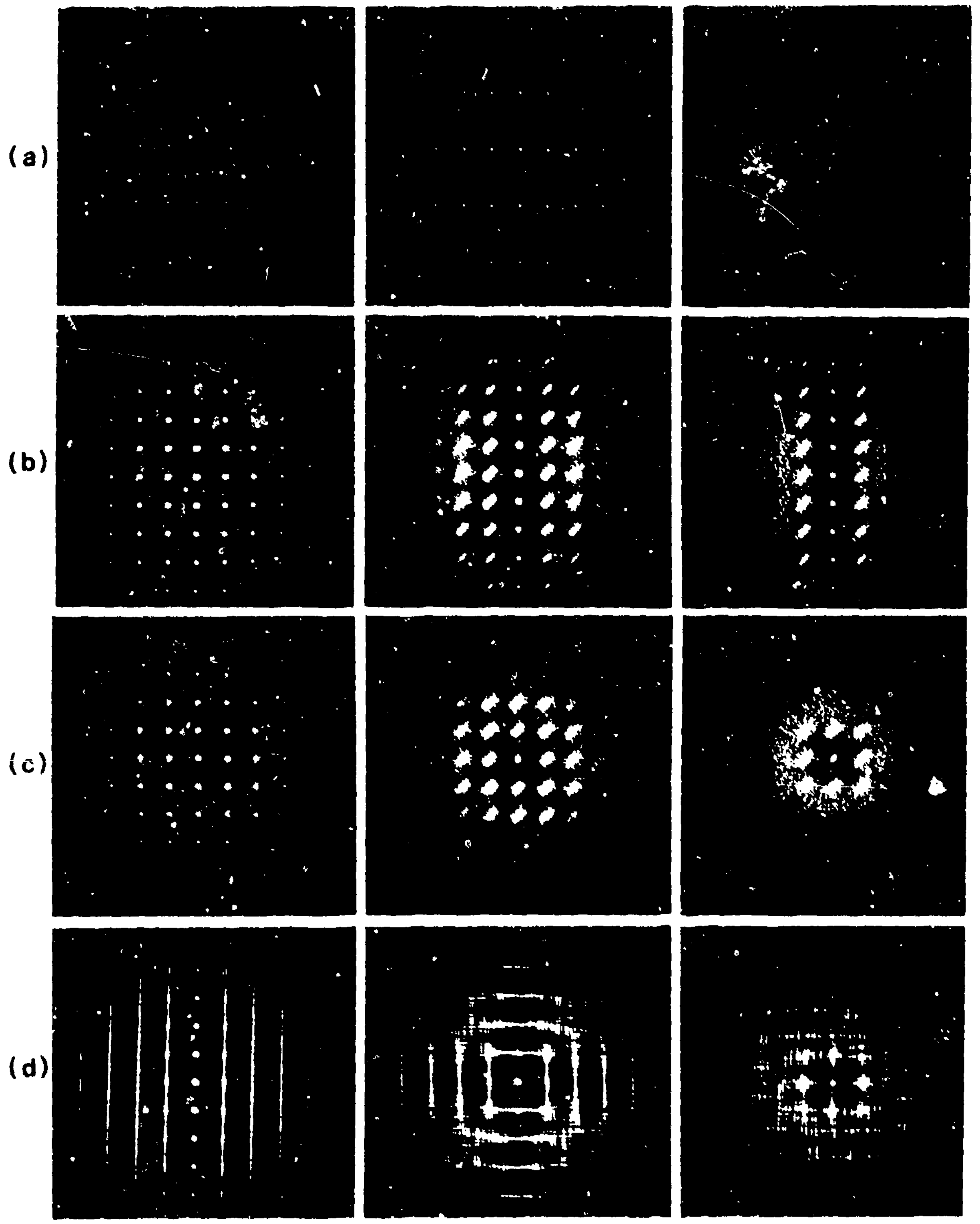
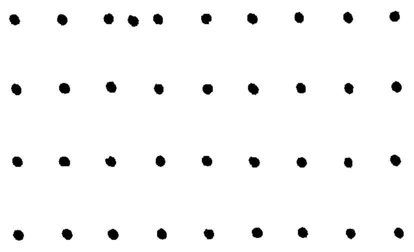
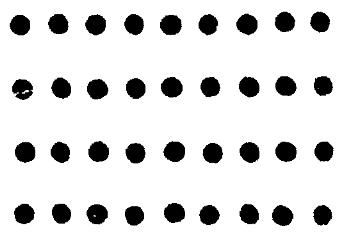


Figure 3.15 Fourier transforms of lattice patterns.

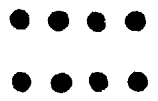
(a) Circular dot pattern



(b) Portion of infinite impulse train



(c) Result of convolving dot with impulse train



(d) Result of multipling (c) by aperture function.

Figure 3.16. Steps in the formation of a lattice pattern.

is also composed of the transform of the dot C(u,v) copied to each impulse location of the sampling impulse function

$$I(u,v) = \sum_{-\infty}^{\infty} \sum_{-\infty}^{\infty} \delta(u-m/X, v-n/Y)$$

which may be written as

$$S(u,v) = \sum_{-\infty}^{\infty} \sum_{-\infty}^{\infty} C(u-m/X, v-n/Y) \quad .$$

Finally, the multiplication of the spatial pattern by the aperature function a(x,y) results in the convolution of the sampled spectra S(u,v) by the aperature transform A(u,v), where

$$A(u,v) = \text{sinc}(\pi u X_1)\text{sinc}(\pi v Y_1)$$

and

$$\text{sinc}(\pi u X_1) = \sin(\pi u X_1)/(\pi u X_1) \quad .$$

A close examination of the transform pattern would reveal the following facts. The shape of the small replica spectra are basically A(u,v). The large overall circular pattern is due to the dot transform C(u,v). The sample spacing is basically due to I(u,v). The overall pattern is described by

$$T(u,v) = A(u,v)*\{I(u,v)\cdot C(u,v)\} \quad .$$

Note that the convolution could be avoided in practice by simply using a finite window. Thus to produce T(u,v) one would simply form C(u,v) and multiply it by I(u,v). The reciprocal effect of the sample spacing is clearly shown in Figure 3.14(b), and in its transform in Figure 3.15(b). Note that a hexagonal dot pattern shown in Figure 3.14(c) has a corresponding hexagonal central portion in its transformation, which is shown in Figure 3.15(c). These examples of perfectly spaced patterns show the basic transform steps involved but would normally not be considered as textures because no irregularity is present. However, consider the pattern shown in Figure 3.14(d).

This pattern is basically the same as Figure 3.14(a), but each point has been displaced both horizontally and vertically from its original amount by an arbitrary amount of up to 10% of the lattice spacing. This effect could be produced by forming a random impulse train i(x,y) where

$$i'(x,y) = \sum_{-\infty}^{\infty} \sum_{-\infty}^{\infty} \delta\{x-m(X+x'),y-nY\}$$

where x' is a uniformly distributed random variable over the normalized interval (-.05,.05). Since $i'(x,y)$ is now a random function, its direct Fourier transform is now not well defined, although the Fourier transform of any particular realization such as that shown in Figure 3.15(d) may certainly be computed. In fact, note that $I'(u,v)$ may also be considered as a random function described by

$$I'(u,v) = \sum_{-\infty}^{\infty} \sum_{-\infty}^{\infty} \delta\{u-m/(X+x'),v-n/Y\} .$$

The variation in the transform spacing would have a normalized range {-1/(.05),1/(.05)}. Thus, if the spatial pattern varies by 10%, the transform pattern could vary by 10 times. This effect can be observed on the transform shown in Figure 3.15(d). A similar effect has been used to produce Figure 3.15(e), with a registration variation between rows and Figure 3.15(f), with 25% variations.

The pattern shown in Figure 3.14(g) may be called a paracrystal. The lattice points are still confined to rows and columns, but both the spacings within rows and columns are subject to 10% variations about the mean value. In Figure 3.14(h), a registration variation between corresponding points in adjacent rows and columns has been added. In Figure 3.14(i), this variation has been increased to 25%.

The pattern shown in Figure 3.14(j) has, in each horizontal row, a perfect one dimensional lattice, but the register of each row is subject to a 10% variation relative to the preceding row. Note that in the transform, this has distributed the energy in the vertical direction. The pattern shown in Figure 3.14(k) is made up of parallelograms of equal side length, but with angle variations. Finally, in Figure 3.14(l), both the cell side length and the angles vary.

This sequence of patterns illustrates that going from very regular patterns to textures involves a random process which can be interpreted in either the spatial or spatial frequency domains. Furthermore, if the basic subpattern remains constant, as is the case in Figure 3.14, the essential random is an impulse train in space or spatial frequency. In the spatial frequency construction, this impulse train is simply multiplied by the transform of the basic pattern's transform. Variations in the texture pattern are produced by the variations in the impulse train spacing.

REFERENCES

1. J. F. Blinn, "Geometric Representations in Computer Graphics," SIGGRAPH Tutorial Notes, 1979.

2. I. E. Sutherland and G. W. Hodgman, "Reentrant Polygon Clipping," CACM, January 1974.

3. B. G. Baumgart, "A Polyhedron Representation for Computer Vision," Proc. of NCC, 1975.

4. S. A. Coons, "Surfaces for Computer Aided Design of Space Forms," M.I.T. Tech. Report MAC-TR-4, June 1967.

5. P. E. Bezier, "Example of an Existing System in the Motor Industry: The Unisurf System," Proc. Roy. Soc. (London), Vol. A321, 1971, pp. 207-218.

6. R. L. Cohen, Lyche, and Riesenfeld, "Discrete B-Splines," Computer Graphics and Image Proc., 1980.

7. L. G. Roberts, "Machine Perception of Three Dimensional Solids," in Optical and Electro-Optical Information Processing, Tippet, et. al., Eds. Cambridge: MIT Press, 1963.

8. R. A. Weiss, "BE VISION, A Package of IBM 7090 FORTRAN Programs to Draw Orthographic Views of Combinations of Plane and Quadric Surfaces," JACM, vol. 13, pp. 194-204, April 1966.

9. A. Appel, "The Notion of Quantitative Invisibility and the Machine Rendering of Solids," Proc. ACM Nat. Conf., pp. 387-393, 1967.

10. I. E. Sutherland, R. F. Sproull, and R. A. Schumacher, "A Characterization of Ten Hidden Surface Algorithms," Computing Surveys, vol. 6, March 1974.

11. M. Bunker and R. Heeschen, "Airborne Electro-Optical Sensor Simulation," AFHRL-TR-75-35, July 1975.

12. M. E. Newell, R. G. Newell, and T. L. Sancha, "A New Approach to the Shaded Picture Problem," Proc. ACM Nat'l. Conf., 1972.

13. R. A. Schumacher, et. al, "Modifications to Interim Visual Spaceflight Fimulator," Final Report, NASA Contract No. NAS9-3916, February 1968.

14. G. S. Watkins, "A Real Time Visible Surface Algorithm," Tech. Report, Univ. of Utah Computer Science Dept. UTECH-CSC-70-101, June 1970.

15. J. F. Blinn, "Models of Light Refelction for Computer Synthesized Pictures," Computer Graphics, vol. 11, Summer 1977.

16. B. T. Phone, "Illumination for Computer Generated Images," Comm. ACM, vol. 18, pp. 311-317, June 1975.

17. K. E. Torrance and E. M. Sparrow, "Theory for Off-Specular Reflection from Roughened Surfaces, JOSA, vol. 57, pp. 1105-1114, Sept. 1967.

18. B. A. Schachter, "Real Time Display of Textures," Proc. 5th Inter. Conf. on Patt. Recog., (Miami), pp. 789-791, Dec. 1980.

19. N. Ahuja and B. Schachter, Pattern Models. (To be published 1981).

20. G. Harburn, C. A. Taylor, and T. R. Welberry, Atlas of Optical Transforms. New York: Cornell University Press, 1975.

4. IMAGE ASSEMBLY SYSTEM COMPONENTS

Since the desirable Image Assembly System capabilities include both image analysis and scene synthesis, it was considered desirable to briefly review individually the characteristics of each type facility separately. Therefore, a typical image analysis facility is described in Section 4.1. Design considerations for a general scene synthesis facility are then reviewed in Section 4.2.

4.1 Image Analysis Facilty

A general-purpose image processing facility should be characterized by three principal features: interactiveness, modularity, and expandability. These features should be reflected in the choice of both the hardware and software elements comprising the system.

This section deals with the basic functional aspects of a proposed image processing facility capable of handling initial processing requirements as well as long term analysis and processing techniques that will result from advances in the state of the art in digital scene analysis.

The principal components of the proposed image processing laboratory are shown in Figure 4.1. The elements of the system are:

Digital computer
Image display system
Video image digitizer
Microdensitometer
Analog video recorder
Disk unit(s)
Digital tape unit
Graphics terminal
Other Input-Output (I/O) devices.

The function and basic requirements of these components are discussed in the following sections.

4.1.1 Digital Computer

The choice of a digital computer is the most important decision to be made in the design and implementation of this facility. The computer hardware sets the limits on the speed at which information within this facility can be processed. The degree of limitation which is placed on the processing of images is directly related to the computational speed of the digital computer. The limitations on the computer hardware must be understood in the context of image processing applications. As the number of computations increases due to algorithm complexity or size of an image, the speed with which the computations are made becomes more critical.

There are three principal features of computer hardware that must be quantified and studied with respect to their effect on the processing of image data. They are: processor speed, maximum core size, and memory addressability. The digital computer selected for this application should have the capability of processing a floating point multiplication in less than one microsecond and should be configured with a minimum of 256K words of memory, with the capability for addressing this size of memory. This suggestion is based on our experience with a large number of image processing algorithms implemented on general-purpose systems. For example, 128K words of memory in a 32-bit machine are required to store two 512 x 512 images with an intensity resolution of 256 levels. Additional storage is required for the operating system, processing algorithms, and intermediate computational results. Although less memory could be used by storing an image on a mass storage device such as a disk, the speed and, hence, the time for processing an image are greatly enhanced by having the capability of storing at least one full frame in core. Additionally, the simplifications in programming that result from having access to the information directly in core is significant.

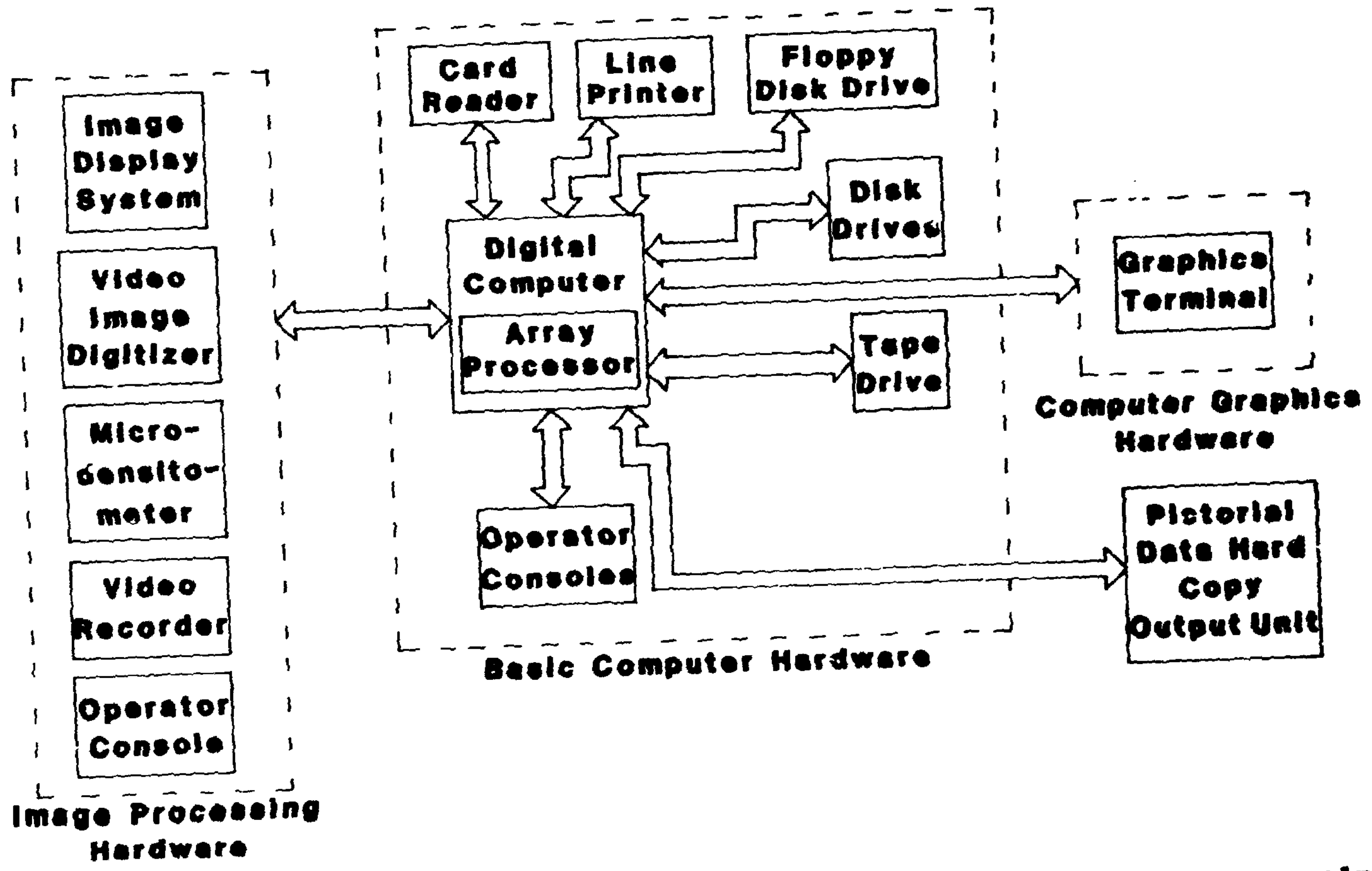


Figure 4.1. Principal components of proposed image processing facility.

The choice of a particular machine should also be influenced by its expandability characteristics and by the selection made at other image processing facilities in the U. S. It is well known that software costs often exceed the cost of hardware, especially in environments in which new techniques are routinely being developed. Similarity of the main processing system will simplify implementation of computer algorithms developed at other institutions.

Addition of an array processor to the system should also be given consideration in order to further reduce computation time. An array processor is particularly important when a large number of images must be analyzed in a real-time environment.

4.1.2 Image Display System

The purpose of the display system is to convert a digitized array back into an analog image. The most popular type of image display system is the memory-refresh type, which produces a video image at 30 frames/sec. from an array stored in dynamic random-access memory.

An important consideration in the choice of an image display system is expanoability in the field. Many manufacturers offer a basic main frame which can accommodate up to three or more channels of 512 x 512, 8-bit images. This allows displays of so-called full-color images by feeding three independent intensity images into the three guns of a color monitor. In terms of the proposed facility, the minimum configuration should be a system capable of displaying 512 x 512 images. With respect to intensity resolution, the minimum number of gray-levels should be 256 (8-bits) in order to capture the subtle changes that are generally characteristic of images used in military systems.

Finally, the availability of a compatible computer interface and well-documented display software written specifically for the operating system selected for the image processing facility should play an important role in the selection of the display system.

4.1.3 Image Digitizer

Image digitizers are used in two basic modes: on-line during testing of sensors and off-line for additional processing of imagery as well as algorithm development. In either case, it is necessary to use an image digitizer in order to convert the analog inputs into a digital representation suitable for computer processing.

The principal requirement of the image digitizer for on-line operation is that it be real time. It is recommended that any digitizer selected for the proposed system be capable of processing a 512 x 512 field with a minimum intensity resolution of 8-bits. The 512 x 512 format is important in applications involving image transforms, which typically require that the number of points in both spatial directions be an integer power of two.

The digitizer used for off-line processing should be able to operate from one of three-input sources: video from a sensor or recorder, transparencies, and printed images. If images are digitized from a recorded video signal, the digitzer used in off-line operation must also be real-time. In the off-line mode, it is important that the digitizer be equipped with reasonable interactive features such as the capability to digitize a window in the image, automatic video gain control, and variable image size. These features should be specifiable either manually or by software control from the computer.

The off-line unit should be physically located in a station containing a camera and light table stand assembled in a permanent arrangement. Off-line operation will be highly interactive, requiring that the digitizer controls be in the vicinity of the camera itself.

4.1.4 Microdensitometer

Microdensitometers are used to produce digital images from hardcopy inputs. Unlike the video digitizer discussed in the previous section which employs the scanning electron beam of a TV camera, a microdensitometer is a device that operates on the principle of quantizing light passing through a transparency or reflected from a nontransparent medium such as a photograph. Microdensitometers employ mechanical scanning to achieve a degree of accuracy that is well beyond the accuracy that can be obtained from scanning with an electron beam. The principal disadvantage of microdensitomers is that they are slow not only in terms of set-up time (i.e., mounting the transparency on a drum or flat bed), but also in terms of scanning speed.

If a microdensitometer is selected for the proposed image processing facility, there are two important points to keep in mind. First, the system should be capable of being interfaced directly to the computer in order to simplify its use. Second, the mounting mechanism and expected size of transparencies should be size compatible in order to avoid having to trim a film so that it will fit in the digitzer. Although this might seem a minor point, it can be a significant source of frustration and inefficiency, often leading to minimum use of the microdensitometer.

4.1.5 Analog Video Recorder

It is recommended that at least one video recorder be included in the image processing facility to record the results of tests deemed particularly important. The video tapes can be used to play back the results of a test for additional processing, evaluation of new algorithms, or for training new operators in the use of the image processing equipment.

The recorder should be of sufficient quality to avoid distorting the data by nonlinear effects. Consideration should be given to having two machines with editing capabilities. This feature will be quite valuable in preparing experimental video tapes containing representative or important events. An editing capability, coupled with audio, also has the advantage of simplifying the generation of video tapes for demonstration purposes.

4.1.6 Disk Units

The principal mass storage of any modern image processing facility is in the form of digital disks. Typically, it is desirable to have at least two independent disk units in order to reduce down-time caused by disk-drive failures. It has been our experience that a capacity on the order of 150 Megabytes per drive will be quite adequate for most image processing facilities, especially in the initial stages of operation. It would be desirable, if bids are competitive, to select disk drives manufactured by the same vendor providing the computer. This will greatly simplify any service agreements that may be purchased for the main components of the system.

4.1.7 Digital Tape Units

Digital tapes will be used primarily for operating software back-up, as well as for archival storage of images. Tapes are also the main medium for off-line data transfer between different computer facilities.

It is anticipated that the image processing facility will not need more than one tape drive. The principal features of this unit should be that it be industry compatible and that it be able to read, as a minimum, a tape density of 1600 bpi (bits per inch).

As indicated in the previous section regarding disk units, it would be advantageous to obtain the tape unit from the same manufacturer as the computer.

4.1.8 Graphics Terminals

The image processing laboratory should contain at least one high-resolution (i.e., 4096 x 4096 points) graphics unit. This unit can also be used for two- and three-dimensional simulation of images, as well as for a large variety of data output display formats.

Although a monochrome-intensity device will suffice for the purposes of graphics and data displays, a strong case can be made for the selection of a color graphics unit. It is well known that the human visual system has the capability of discerning several orders of magnitude more colors than monochrome intensities. The accuracy of data analysis and interaction of a human and a machine are often enhanced by the use of color. Also, summary of results for presentations are considerably more effective when prepared in color.

4.1.9 Other Input-Output (I/O) Devices

The image processing facility will require several console CRT terminals. These terminals will be used in the computer system and the graphics and image display stations, as well as for communication with other computer facilities.

Two basic hard copy output devices will also be needed in the system. The first is a line printer or similar unit for listing programs, numerical outputs, etc. The second is a photographic unit for recording any desired outputs from the image display and graphics systems. The simplest photographic device is a polaroid or 35-mm single-reflex camera. There are available units capable of reproducing TV outputs in a variety of formats, such as photographs, transparencies, and 35-mm slides. These units perform a very important function in an image processing environment by providing fast, high-quality records of experimental results. They also simplify the generation of pictorial materials for reports and presentations.

Additional I/O devices that should be considered for the system include a dual floppy disk drive and a card reader. Although these last two items are the least important, they should be evaluated in terms of projected uses of the facility.

4.1.10 A Note on Software Documentation

A procedure for documenting all software developments and a standard image file format should be established prior to the date when the image processing system is expected to begin operation. Although this may appear as a trivial recommendation, an informal tour through other image processing facilities would soon reveal a prevalent lack of proper documentation, poorly written software, and incompatible image formats.

All software should be written using structured programming techniques. A User Note should be written for every program that is made an integral part of the image processing package. As a minimum, this note should include:

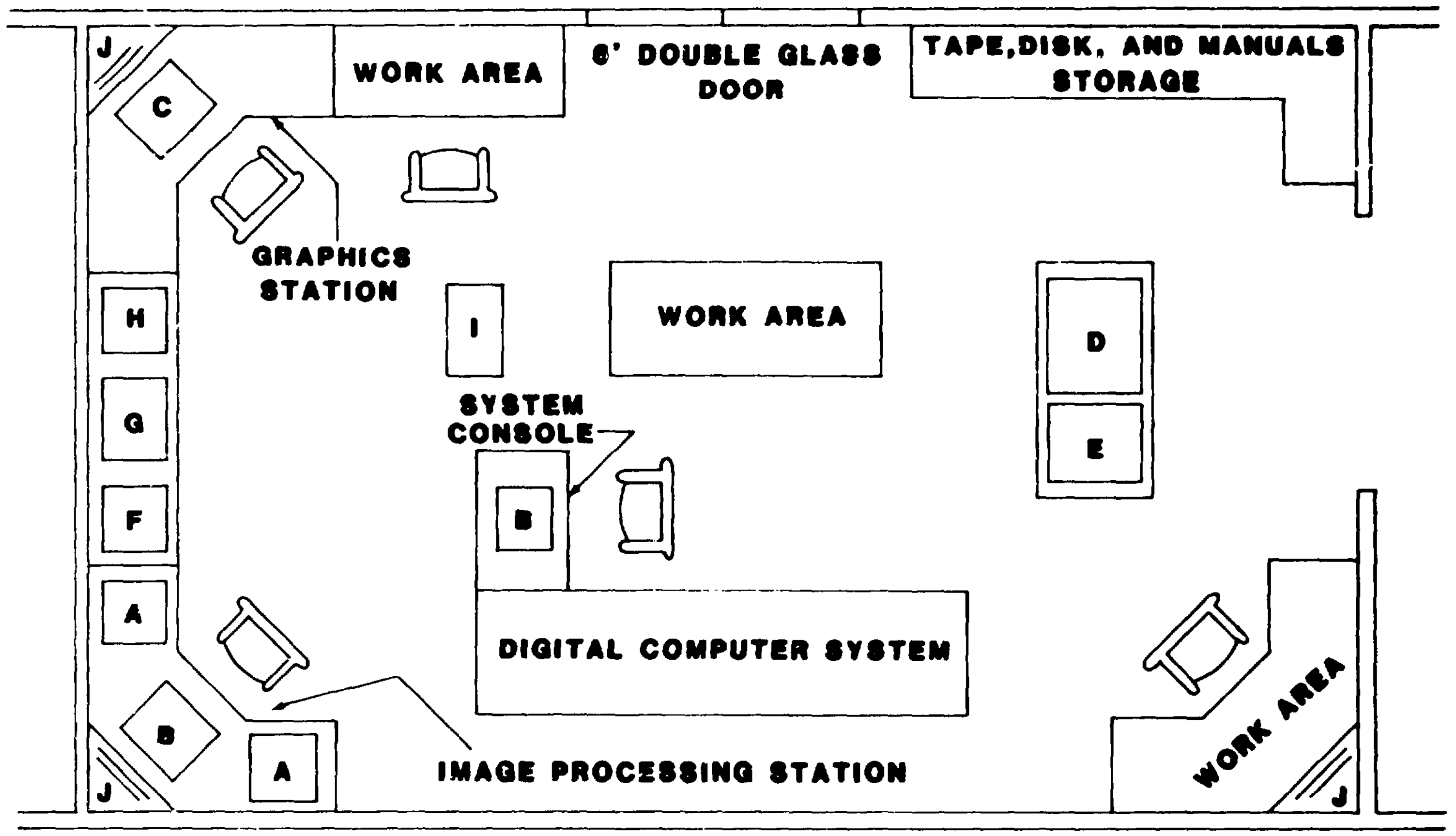
1. the date
2. the note number in an established sequence
3. the name of the programmer
4. a clear set of instructions using standard terminology with respect to other notes and detailing all options available in the program
5. a reference to the location of the original program source.

All image files should consist of two basic components: a header section and an image data section. The header section of the image file will serve as an identifier and will play a central role in data organization and retrieval. This particular format represents one of the first attempts at standardizing image data, and it appears to be gaining acceptance in this country, as well as in Europe.

The data section of the image file should be divided into records whose size is chosen to optimize data transfers. Choice of record size will have to be delayed until the hardware has been selected.

4.1.11 Organization of the Image Processing Laboratory

Figure 4.2 shows a suggested layout of the image processing



A - TV MONITORS
B - COMPUTER CONSOLE
C - GRAPHICS CONSOLE
D - LINE PRINTER
E - CARD READER
F - VIDEO IMAGE DIGITIZER
G - MICRODENSITOMETER
H - VIDEO RECORDER
I - PICTORIAL DATA OUTPUT UNIT
J - SHELVES

Figure 4.2. Layout for the Image Processing Laboratory.

SCALE:1/4"II1'.0"

laboratory in a space adequate for this facility. It is expected that a raised tile floor will be installed during construction in order to guarantee adequate heat removal and to simplify interconnection of the equipment.

In terms of lighting, it is recommended that a dual lighting system be installed. One system should be of the fluorescent type normally used in offices. The other should consist of incandescent spotlights with dimmers. Both lighting systems should be partitioned into sections with independent controls to allow variable lighting configurations. The spotlights are particularly important in the area where the image processing and graphics stations are located. It has been our experience that soft, incandescent lights are ideally suited for the type of work that will be carried out at these stations.

Careful consideration should be given to a planned, easily accessible storage facility for tapes, disks, and manuals. This can be accomplished by the arrangement shown in Figure 4.2 and by strategically placed shelves around the room.

4.2 Image Synthesis Facility

4.2.1 Functional Specification of Software for a Scene Synthesis System

A scene synthesis system may be divided into hardware and software. The question which will be addressed in this section is: Can a functional specification be written for the software required in a scene synthesis system? Perhaps one should first consider why a functional specification of software is desirable. The answer is yes if portability is desirable. Another natural question is whether it is possible to specify software requirements. The development of the ACM Core System standards for computer graphics software provides evidence that it is possible. This standard will now be briefly reviewed.

The ACM Core System describes a methodology for graphics standards design and presents a standard graphics package designed according to this methodology. The methodology covers both vector graphics and shaded raster graphics. Image processing is not included.

A fundamental motivation for the standard was the desire for probgram portability or the ability to transport graphics applications from one laboratory to another with minimal program changes. Three levels of portability were considered:

1. No source program modifications at all.
2. Modifications of a purely editorial nature, such as the substitution of LINE for DRAW.
3. Modifications to the structure of the program which are usually difficult and fraught with errors.

With a standard, it is possible for some programs to be transported without any source program changes. Also, with a standard, it is possible to obtain programmer portability or the ability for an applications programmer to move from one installation to another with minimal retraining.

The simple model used by the ACM for the graphics package specification is shown in Figure 4.3. Note that this model is appropriate for the scene synthesis system. The following concepts are also considered fundamental:

1. The separation of input and output functions.
2. The minimization of the differences between producing output on a hard copy device such as a plotter and on an interactive display.
3. The concept of two coordinate systems; the world coordinate system in which the scene is constructed, and the device coordinate system in which the picture is displayed.
4. The concept of a display file containing device coordinate information, used by all but the least interactive of graphics systems.
5. The idea of display file segments, mutually independent, each of which can be modified as a unit.
6. The provision of functions to transform world coordinate data into device coordinates called a viewing operation.

It is also recognized in the ACM standard that the previous model may prove unsatisfactory under certain conditions, either because it lacks capabilities or because components must be included which degrade performance. The proposed solution is to seek carefully designed modularity in the graphics package and levels of modularity such that all systems could be compatible at the highest level but not necessarily

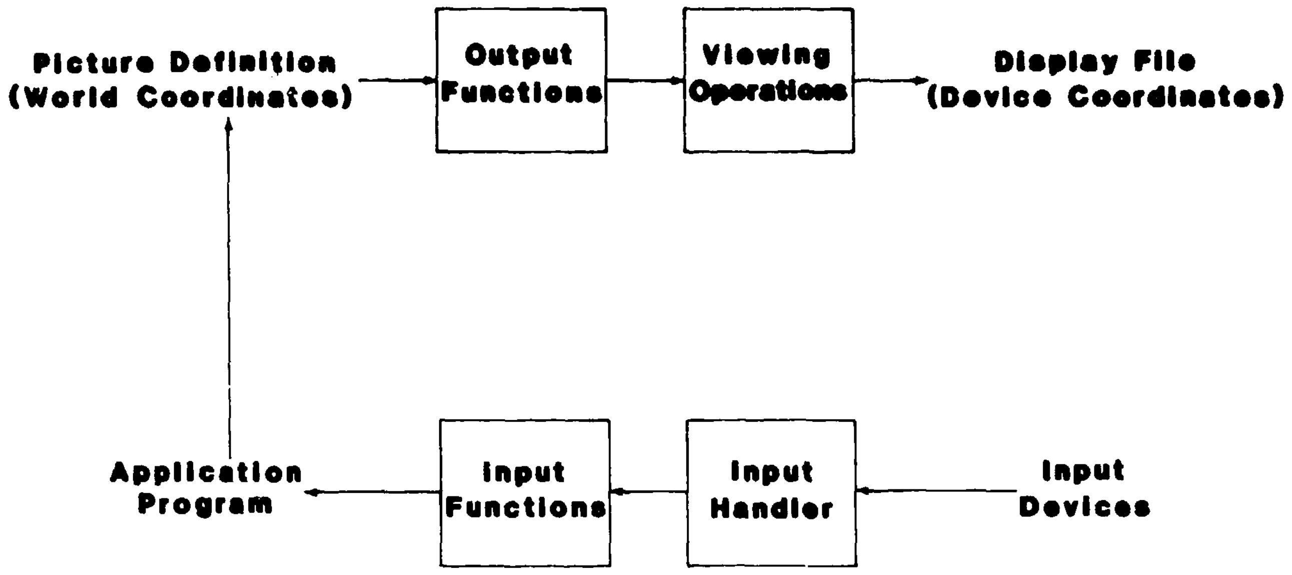


Figure 4.3. Structure of a graphics package.

at the lowest level. Some of the high level modules are:

1. A symbol system, providing functions for defining sub-pictures and for inserting instances of these sub-pictures into the world coordinate scene definition.
2. A high quality text system.
3. A curve display system.
4. Higher level plotting and map-making modules.

The full functional specification of the ACM report is very extensive and may be found in [1]. The function listing from the specification is shown in Table 4.1. The large number of functions is to some extent the result of the rich variety of graphics hardware devices such as storage, refresh, and color displays and graphics tablets, light pens, and function keyboards for input. There are also natural variations of basic functions which lead to several entries in the table--for example, whether a position is specified in an absolute coordinate system, ABS, or relative to the current position, REC, gives rise to duplicate entries such as MOVE ABS and MOVE REL. However, the table is structured so that groups of functions required for a particular type system are easily identified. The functions in Table 4.1 are important for a general purpose graphics system; however, an extension to the standard, called the Raster Extension, is much more germane to a scene synthesis system.

4.2.2 Raster Extension

The Raster Extension to the ACM Core System was designed to extend the concept of a portable device-independent graphics subroutine package for use with raster, refresh, or image type displays. The raster display model was taken to include the following:

1. A display may have more than one refresh buffer, either physically or logically distinct.
2. The (x,y) size of the refresh buffer may be larger than can be displayed at one time.
3. The z size of the refresh buffer varies over a wide range from 1 to 24, with 8 or less being typical.
4. The display may include one or more look-up tables used to generate color or gray scale images.
5. The raster display consists mostly of memory both in its refresh buffers and its look-up tables.
6. The display is typically connected to a host computer by a high speed line.
7. Each pixel value is typically calculated by the host computer, although hardware subsystems which draw lines, text, and figures are becoming more common.
8. Pixels may be both written and read from the refresh memories.

Note that the model encompasses a wide range of existing equipment, such as the Tektronix 4027 or a Comtal Vision 120. The range of the model covers the raster display, raster terminals, and even computer output to microfilm devices.

The functional capabilities included in the raster extension include:

1. filled in polygonal areas;
2. extensive color and intensity specifications;
3. display of computer or terminal generated patterns, and;
4. consideration of the hidden surface removal problem.

Specific recommendations are included for polygon shading, primitive attributes, static attributes, initialization and termination, view surface initialization and termination, picture change control, color specifications, pixel array functions, and index table functions.

The Raster Extension was only introduced in 1979. Thus, certain revisions must be expected before the standard becomes well defined. However, it does address the scene synthesis area specifically and, since it is developed by the professional organization in this area, it probably should be seriously considered. The Core System is referenced in one of the latest texts [2] but not specifically adhered to in the text. Most commercial systems can be expected to use the standard at least in new systems.

Output Primitives:

Basic, None, 2D

- MOVE_ABS_2
- MOVE_REL_2
- INQUIRE_CURRENT_POSITION_2
- LINE_ABS_2
- LINE_REL_2
- POLYLINE_ABS_2
- POLYLINE_REL_2
- TEXT
- INQUIRE_TEXT_EXTENT_2
- MARKER_ABS_2
- MARKER_REL_2
- POLYMARKER_ABS_2
- POLYMARKER_REL_2

Basic, None, 3D

- MOVE_ABS_3
- MOVE_REL_3
- INQUIRE_CURRENT_POSITION_3
- LINE_ABS_3
- LINE_REL_3
- POLYLINE_ABS_3
- POLYLINE_REL_3
- INQUIRE_TEXT_EXTENT_3
- MARKER_ABS_3
- MARKER_REL_3
- POLYMARKER_ABS_3
- POLYMARKER_REL_3

Picture Segmentation and Naming:

Basic, None, 2D

- CREATE_TEMPORARY_SEGMENT
- CLOSE_TEMPORARY_SEGMENT
- INQUIRE_OPEN_TEMPORARY_SEGMENT

Buffered, None, 2D

- CREATE_RETAINED_SEGMENT
- CLOSE_RETAINED_SEGMENT
- DELETE_RETAINED_SEGMENT
- DELETE_ALL_RETAINED_SEGMENTS
- RENAME_RETAINED_SEGMENT
- INQUIRE_RETAINED_SEGMENT_SURFACES
- INQUIRE_RETAINED_SEGMENT_NAMES
- INQUIRE_OPEN_RETAINED_SEGMENT

Attributes:

Basic, None, 2D

- SET_COLOR
- SET_INTENSITY
- SET_LINESTYLE
- SET_LINEWIDTH
- SET_PEN
- SET_FONT
- SET_CHARSIZE
- SET_CHARUP_2
- SET_CHARPATH
- SET_CHARSPACE
- SET_CHARJUST
- SET_CHARPRECISION
- SET_MARKER_SYMBOL
- SET_PICK_ID
- SET_PRIMITIVE_ATTRIBUTES_2
- INQUIRE_COLOR
- INQUIRE_INTENSITY
- INQUIRE_LINESTYLE
- INQUIRE_LINEWIDTH
- INQUIRE_PEN
- INQUIRE_FONT
- INQUIRE_CHARSIZE
- INQUIRE_CHARUP_2
- INQUIRE_CHARPATH
- INQUIRE_CHARSPACE
- INQUIRE_CHARJUST
- INQUIRE_CHARPRECISION
- INQUIRE_MARKER_SYMBOL
- INQUIRE_PICK_ID
- INQUIRE_PRIMITIVE_ATTRIBUTES_2

Table 4-1. ACM Core Functions

```
Basic, None, 3D
     SET_CHARPLANE
     SET_CHARUP_3
     SET_PRIMITIVE_ATTRIBUTES_3
     INQUIRE_CHARPLANE
     INQUIRE_CHARUP_3
     INQUIRE_PRIMITIVE_ATTRIBUTES_3
Buffered, None, 2D
     SET_VISIBILITY
     SET_HIGHLIGHTING
     INQUIRE_VISIBILITY
     INQUIRE_HIGHLIGHTING
     SET_SEGMENT_VISIBILITY
     SET_SEGMENT_HIGHLIGHTING
     INQUIRE_SEGMENT_VISIBILITY
     INQUIRE_SEGMENT_HIGHLIGHTING
Buffered, Synchronous, 2D
     SET_DETECTABILITY
     INQUIRE_DETECTABILITY
     SET_SEGMENT_DETECTABILITY
     INQUIRE_SEGMENT_DETECTABILITY
Dynamic-a, None, 2D
     SET_IMAGE_TRANSFORMATION_TYPE
     INQUIRE_IMAGE_TRANSFORMATION_TYPE
     INQUIRE_SEGMENT_IMAGE_TRANSFORMATION_TYPE
     SET_IMAGE_TRANSLATE_2
     INQUIRE_IMAGE_TRANSLATE_2
     SET_SEGMENT_IMAGE_TRANSLATE_2
     INQUIRE_SEGMENT_IMAGE_TRANSLATE_2
Dynamic-b, None, 2D
     SET_IMAGE_TRANSFORMATION_2
     INQUIRE_IMAGE_TRANSFORMATION_2
     SET_SEGMENT_IMAGE_TRANSFORMATION_2
     INQUIRE_SEGMENT_IMAGE_TRANSFORMATION_2
Dynamic-c, None, 3D
     SET_IMAGE_TRANSFORMATION_3
     INQUIRE_IMAGE_TRANSFORMATION_3
     SET_SEGMENT_IMAGE_TRANSFORMATION_3
     INQUIRE_SEGMENT_IMAGE_TRANSFORMATION_3

           Viewing Operations and Coordinate Transformations

Basic, None, 2D
     SET_WINDOW
     SET_VIEW_UP_2
     SET_NDC_SPACE_2
     SET_VIEWPORT_2
     INQUIRE_WINDOW
     INQUIRE_VIEW_UP_2
     INQUIRE_NDC_SPACE_2
     INQUIRE_VIEWPORT_2
     MAP_NDC_TO_WORLD_2
     MAP_WORLD_TO_NDC_2
     SET_WINDOW_CLIPPING
     INQUIRE_VIEWING_CONTROL_PARAMETERS
     SET_WORLD_COORDINATE_MATRIX_2
     INQUIRE_WORLD_COORDINATE_MATRIX_2
Basic, None, 3D
     SET_VIEW_REFERENCE_POINT
     SET_VIEW_PLANE_NORMAL
     SET_VIEW_PLANE_DISTANCE
     SET_VIEW_DEPTH
     SET_PROJECTION
     SET_VIEW_UP_3
     SET_NDC_SPACE_3
     SET_VIEWPORT_3
     SET_VIEWING_PARAMETERS
     INQUIRE_VIEW_REFERENCE_POINT
     INQUIRE_VIEW_PLANE_NORMAL
     INQUIRE_VIEW_PLANE_DISTANCE
     INQUIRE_VIEW_DEPTH
     INQUIRE_PROJECTION
     INQUIRE_VIEW_UP_3
     INQUIRE_NDC_SPACE_3
     INQUIRE_VIEWPORT_3
     INQUIRE_VIEWING_PARAMETERS
     MAP_NDC_TO_WORLD_3
     MAP_WORLD_TO_NDC_3
     SET_FRONT_PLANE_CLIPPING
     SET_BACK_PLANE_CLIPPING
```

Table 4-1. Continued

```
        SET_COORDINATE_SYSTEM_TYPE
        SET_WORLD_COORDINATE_MATRIX_3
        INQUIRE_WORLD_COORDINATE_MATRIX_3
             Input Primitives:
Basic, Synchronous, 2D
        INITIALIZE_DEVICE
        INITIALIZE_GROUP
        TERMINATE_DEVICE
        TERMINATE_GROUP
        AWAIT_ANY_BUTTON
        AWAIT_KEYBOARD
        AWAIT_STROKE_2
        AWAIT_ANY_BUTTON_GET_LOCATOR_2
        AWAIT_ANY_BUTTON_GET_VALUATOR
        SET_ECHO
        SET_ECHO_GROUP
        SET_ECHO_SEGMENT
        SET_ECHO_SURFACE
        SET_ECHO_POSITION
        SET_KEYBOARD
        SET_BUTTON
        SET_ALL_BUTTONS
        SET_STROKE
        SET_LOCATOR_2
        SET_LOCPORT_2
        SET_VALUATOR
        INQUIRE_INPUT_CAPABILITIES
        INQUIRE_INPUT_DEVICE_CHARACTERISTICS
        INQUIRE_ECHO
        INQUIRE_ECHO_SURFACE
        INQUIRE_ECHO_POSITION
        INQUIRE_KEYBOARD
        INQUIRE_BUTTON
        INQUIRE_STROKE
        INQUIRE_ECHO_SEGMENTS
        INQUIRE_LOCATOR_2
        INQUIRE_LOCPORT_2
        INQUIRE_VALUATOR

Basic, Synchronous, 3D
        AWAIT_STROKE_3
        AWAIT_ANY_BUTTON_GET_LOCATOR_3
        SET_LOCATOR_3
        SET_LOCPORT_3
        INQUIRE_STROKE_DIMENSION
        INQUIRE_LOCATOR_DIMENSION
        INQUIRE_LOCATOR_3
        INQUIRE_LOCPORT_3

Basic, Complete, 2D
        ENABLE_DEVICE
        ENABLE_GROUP
        DISABLE_DEVICE
        DISABLE_GROUP
        DISABLE_ALL
        READ_LOCATOR_2
        READ_VALUATOR
        AWAIT_EVENT
        FLUSH_DEVICE_EVENTS
        FLUSH_GROUP_EVENTS
        FLUSH_ALL_EVENTS
        ASSOCIATE
        DISASSOCIATE
        DISASSOCIATE_DEVICE
        DISASSOCIATE_GROUP
        DISASSOCIATE_ALL

        GET_KEYBOARD_DATA
        GET_STROKE_DATA_2
        GET_LOCATOR_DATA_2
        GET_VALUATOR_DATA
        INQUIRE_DEVICE_STATUS
        INQUIRE_DEVICE_ASSOCIATIONS

Basic, Complete, 3D
        READ_LOCATOR_3
        GET_STROKE_DATA_3
        GET_LOCATOR_DATA_3
```

Table 4-1. Continued

```
Buffered, Synchronous, 2D
     AWAIT_PICK
     SET_PICK
     INQUIRE_PICK

Buffered, Complete, 2D
     GET_PICK_DATA

          Control:

Basic, None, 2D
     INITIALIZE_CORE
     TERMINATE_CORE
     INITIALIZE_VIEW_SURFACE
     TERMINATE_VIEW_SURFACE
     INQUIRE_OUTPUT_CAPABILITIES
     SELECT_FOR_SEGMENT_CONSTRUCTION
     DESELECT_FOR_SEGMENT_CONSTRUCTION
     INQUIRE_SELECTED_SURFACES
     SET_IMMEDIATE_VISIBILITY
     MAKE_PICTURE_CURRENT
     BEGIN_BATCH_OF_UPDATES
     END_BATCH_OF_UPDATES
     INQUIRE_CONTROL_STATUS
     SET_VISIBILITIES
     NEW_FRAME
     REPORT_MOST_RECENT_ERROR
     LOG_ERROR

          Special Interfaces:

Basic, None, 2D
     ESCAPE
     INQUIRE_ESCAPE
```

Table 4-1. Continued

4.2.3 Hardware Architectures for Scene Synthesis

A review of hardware architectures of past and present scene synthesis systems is presented in this section in order to provide a basis for a forecast of future systems architecture and programming requirements. Since the majority of past work in scene synthesis has been for flight simulation systems, several recent papers in this area are reviewed which provide not only the history of this development, but also the basis for a functional specification of the hardware architecture.

A review of the use of computers in real time simulations of aircraft for training applications was recently presented by Amico and Lindahl. Immediately following World War II, the Office of Research and Invention undertook the development of a number of computer systems. The Whirlwind system was a digital computer development whose initial objective was to simulate the flight envelope of an aircraft to obtain pilot evaluation of proposed new aircraft based upon wind tunnel data before going to the expense of developing a prototype. This system led to a powerful computer for its day and led to a more rigorous mathematical description of the equations of flight.

In the late Forties, the flight simulators developed for the Armed Services called for the solution of the differential equation of motion using analog computer systems. Three major difficulties were encountered in the use of analog computers for flight simulators:

1) the design of the computer was constrained by the availability and accuracy of aerodynamics and system data on the aircraft;

2) once the system was built and testing revealed errors in the aerodynamic data, the computer hardware design had to be changed;

3) the errors and limitations of analog systems led to serious issues with users.

These difficulties motivated and still motivate the use of digital technology. The expected advantages of digital technology were:

a) that programming would be independent of the computer and input/output system design; and

b) changes in aircraft aerodynamic data could be incorporated without hardware changes to the computer system in most cases.

Interestingly, the Whirlwind did not meet its initial objective. The failure of Whirlwind as a real time digital flight simulator triggered an investigation into the reasons for the failure and led to a new research effort to develop the required technology. This study was initiated in 1949, conducted by the Moore School of Engineering at the University of Pennsylvania. The results of the 1950 study concluded that there did not exist a digital computer system capable of solving the flight equations for one aircraft in real time using integration schemes in common use. Later studies in 1952 and 1954 concluded that the real time simulation of aircraft was feasible. The basis for this conclusion was the development of three important advances:

1) high-speed random access memory and digital logic which increased the speed of computation;

2) new algorithms for numerical integration which were consistent with the speed of the digital computer; and

3) time efficient methods of function generation for calculating aerodynamic coefficients and stability derivatives.

In 1956, a contract was awarded under the sponsorship of the Air Force and Navy for the design, development, and construction of a digital computer system in accordance with the logical structure and circuits developed by the Moore School. This effort was known as the Universal Digital Operational Flight Trainer (UDOFT). The purpose of the effort was to demonstrate that a digital computer system could be designed to simulate in real time subsonic and supersonic jet aircraft, utilizing the appropriate cockpit configurations and controls. The benefits derived from the use of a digital computer were improved accuracy, flexibility, and the relative ease of programming. The UDOFT used a general purpose CPU with 5 μsec memory access time, 8 K words of RAM core memory, had no mass storage, and was programmed in machine language. These characteristics and the more recent developments are shown in Table 4.2, from Amico and Lindahl [3]. Note the evolution of all system specifications over the last 20 years. Memory access time has decreased from 5 μsecs to 600 nsecs. Memory capacity has increased from 8 K words to 128 K words. Type of memory has changed from memory core to semiconductor MOS, mass storage has increased from none to 300 megabytes. Programming ease has advanced from machine language through assembly to a high level language, FORTRAN. The number of CPU per Cockpit has increased from 1 to 4 as the distributed processor concept, which minimizes the computer cabling requirement, has evolved. At the present time, research is being conducted in the area of distributed processor computer architectures, which attempt to use the increased computing power and decreased cost of mini and microcomputers.

AIRCRAFT	DEVICE/ TYPE	TIME FRAME	NUMBER & TYPE OF CPU	ACCESS TIME (SEC)	RAM WORDS	TYPE OF RAM	MASS STORE	PROGRAM LANGUAGE	CPU/ COCKPIT	TYPE CONFIG
F9F-2 or F100A	UROFT	1960	1 - SPECIAL PURPOSE DIGITAL COMPUTER	5 μSEC	8 KW	CORE	NONE	MACHINE LANGUAGE	1/1	1
A-7A	2F84 OFT	1964	1 - DDP-224	6 μSEC	32 KW	CORE	NONE	ASSEMBLY	1/1	1
TA-4F	2F90 OFT	1968	2 - SIGMA 5	850 NSEC	8 KW-FLIGHT 16 KW-SYSTEM 8 KW-SHARED	CORE	NONE	ASSEMBLY	2/4	2
T-2C	2F101 OFT	1970	4 - PDP 11/45	850 NSEC	48 KW EACH	CORE	4-1 MB DISC	ASSEMBLY	1/1	1
F-14A	2F95 OFT	1970	1 - SIGMA 5	850 NSEC	32 KW EACH	CORE	1-3 MB DISC	ASSEMBLY	1/1	1
A-4M	2F108 OFT	1972	1 - DATACRAFT 6024/5	1 μSEC	32 KW	CORE	1-0.5 MB DISC	ASSEMBLY	1/1	1
F/A-18	2F132 OFT	1979	4 - SEL 32/77	600 NSEC	128 KW EACH 64 KW-SHARED	MOS	2-300 MB DISC	FORTRAN	4/1	4

Table 4-2. FLIGHT TRAINER COMPUTER SYSTEM CHARACTERISTICS

4.2.4 Display/Iteration Rate

Amico and Lindahl provide a very interesting discussion of the computation iteration and image display rates required for real time simulation of aircraft flight. The initial program designs used a heuristically established iteration rate of 20 Hz. Subsequent investigations established that the aircrafts' natural frequencies were in the range of 1 Hz. Thus the iteration rate was approximately 10 times that required by the Nyquist criterion of two samples per cycle. However, other factors were involved, such as the ability of the pilot to introduce a step input into the system. Some analyses indicated that rates in excess of 1000 Hz may be necessary to achieve stable computations. Other studies and the requirement to minimize synchronization problems with TV systems led to a 30 Hz rate being established by the Navy. Amico and Lindahl report that the selection of the 30 Hz rate seems to have solved the problem.

4.2.5 Programming Language and Software Requirements

Although the initial simulators were programmed in machine and assembly languages, a major trend toward higher level languages is described by Amico and Lindahl. This software explosion is indicated in Figure 4.4, which shows the exponential growth in both RAM and disk storage. Studies conducted for the Air Force and Navy in the early 1970"s into the use of FORTRAN encountered some negative reactions and criticisms, but have led to the now general acceptance of a FORTRAN requirement. However, the trend now is toward the adoption of ADA, the high level language developed for the DOD, when it becomes available.

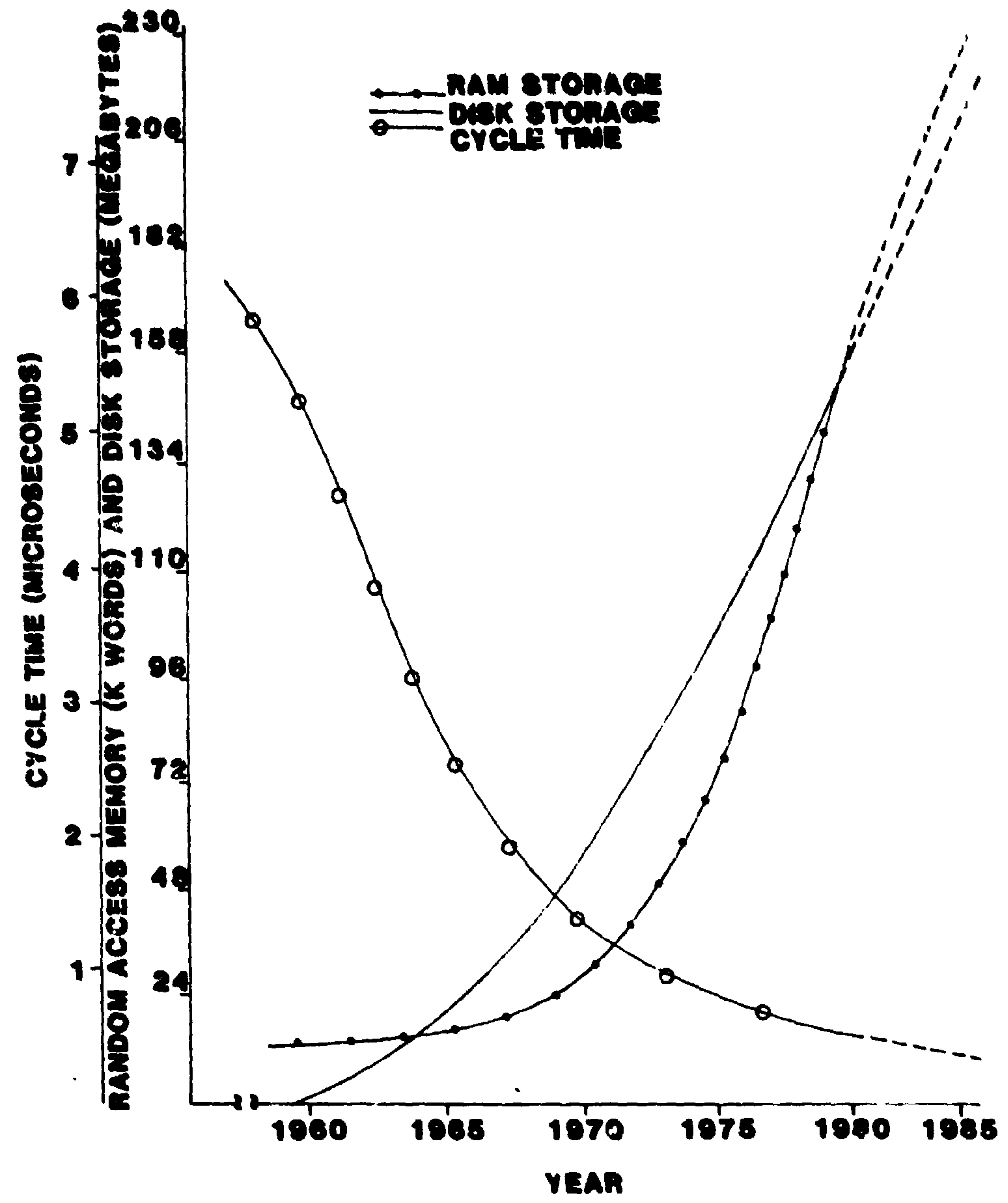


Figure 4.4. Flight Trainer Computing System Trends.

4.2.6 Hardware Architectures

A new visual system architecture recently proposed by Schumaker [4] is shown in Figure 4.5 Schumaker states that simulation of the real world remains an implicit but illusive standard for computer image generation (CIG), and that past successes are driving CIG systems into more complex and comprehensive training applications. He also mentions that the gap between what is feasible and what is desired can only be bridged by understanding both the visual system and the training requirements. The architecture of an image generation system includes the organization, the algorithms, and the implementation methodology and technology. The CT-5 system architecture shown in Figure 4.5 emerged in response to a need for simultaneous display of high image content and quality across a large field of view, requiring many channels. The CT-5 system has been used to display images with a thousand polygons/channel and several thousand polygons total.

The system shown in Figure 4.5 is divided into three major sections: a general purpose computer, a special purpose computer called the image processor, and a display. The image processor is the element which makes the system unique and capable of scene synthesis. The image processor is divided into two processors: the viewpoint and the channel processors.

The viewpoint processor consists of an object manager (OM) and a polygon manager (PM). These elements perform tasks related to the entire scene. The entire system is structured as a pipeline of memory buffers and processors. One television field time is allocated to transport and process data from one memory to the next. The capacity of each section is therefore determined by this processing time. The shaded portion of the polygon manager in Figure

4.5 indicates its field buffer memory. The viewpoint processor also contains two environment memories that provide online storage for portions of the data base sufficiently close to the viewpoint that they may be required for scene generation. In a hierarchical succession of computation steps, the viewpoint processor sifts through the data base to extract a minimal required set of scene elements to produce the current picture. This includes a sequence of increasingly discriminating tests for field of view inclusion, rejection of backfacing elements, and perspective size discrimination. Therefore, the scene data provided to the channel processor is highly refined.

The channel processor provides the computational power to generate a TV format image. The geometric processor (GP) performs the 3D to 2D transformation including rotation, clipping, and perspective. An image plane description of all scene elements that project on a display is stored in the polygon buffer. The remaining steps required to generate an image are performed in the display processor (DP).

The display processor performs the operations necessary to convert image plane descriptions of individual scene elements into a composite picture with hidden surfaces removed, anti-aliasing techniques applied, and TV formatting accomplished. The CT-5 systems abandon the historical E & S scanline approach to this picture computation. Images are computed in feature sequential order rather than scanline order. Area processors replace scanline processors, and area representations replace scanline samplers.

Image processing is performed in parallel hardware which operates on regions in the display called spans. The display plane is divided into a contiguous set of these span areas which may be thought of as a coarse

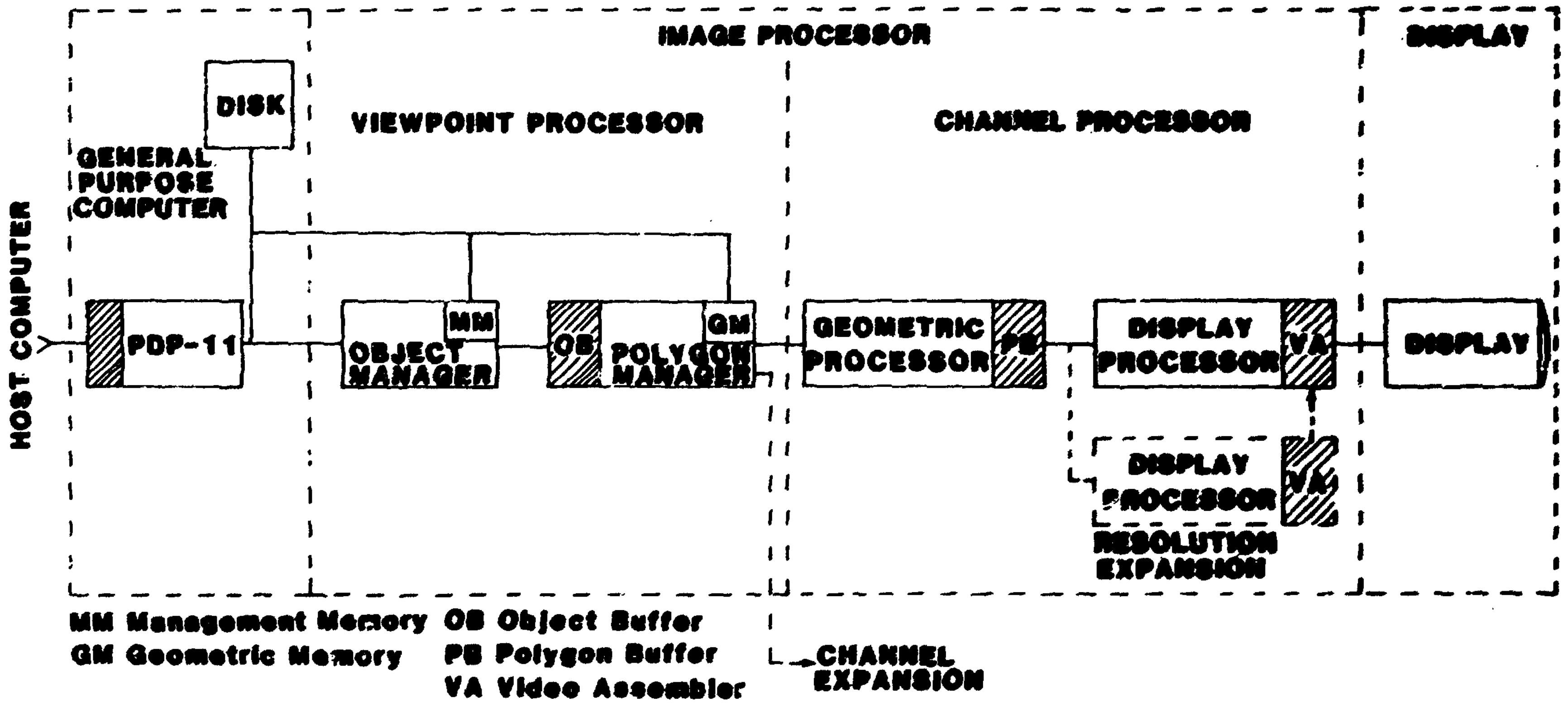


Figure 4.5. Basic CT-5 System Configuration.

pixel set. Individual scene elements are processed a span at a time. Since the scene elements are arranged in order of decreasing visual priority, closer objects occlude farther ones.

The architecture of the CT-5 system includes novel, modular components for the generation of scenes composed of polygons. Parallel area processing is a central theme of the image display processing found in the system.

A different architecture is described by Dichter, et. al. [5], and is shown in Figure 4.6. The system may again be divided into three elements: a general purpose computer, a special image processor, and a display. The special processor consists of a geometric processor, display generator, and illumination control. The system timing is based on a double buffer design so that two TV frame times may be used for processing without interfering with the real time display.

The proposed REALSCAN system, shown in Figure 4.7 and described by Spooner, et. al. [6], utilizes a hierarchy of resolution levels to model the environment and parallel, pipelined processors to perform the visibility determination. The bulk memory in Figure 4.7 is a low cost videodesk which is used to store color and height data for the scene. The control computer determines which blocks of data representing hierarchical areas of terrain in the bulk memory are required for generating the desired scene. These data are transferred to the cell memory. The pipeline processing for image generation is then implemented. The environment model for this system is limited to terrain surfaces which are single valued in elevation.

In the previous architectures, either polygonal or terrain type data bases were accomodated. The problems involved in accomodating both are described by Cunningham and Picasso [7]. Since the type data base determines the architecture to a large extent, systems accomodating both appear most promising for the future.

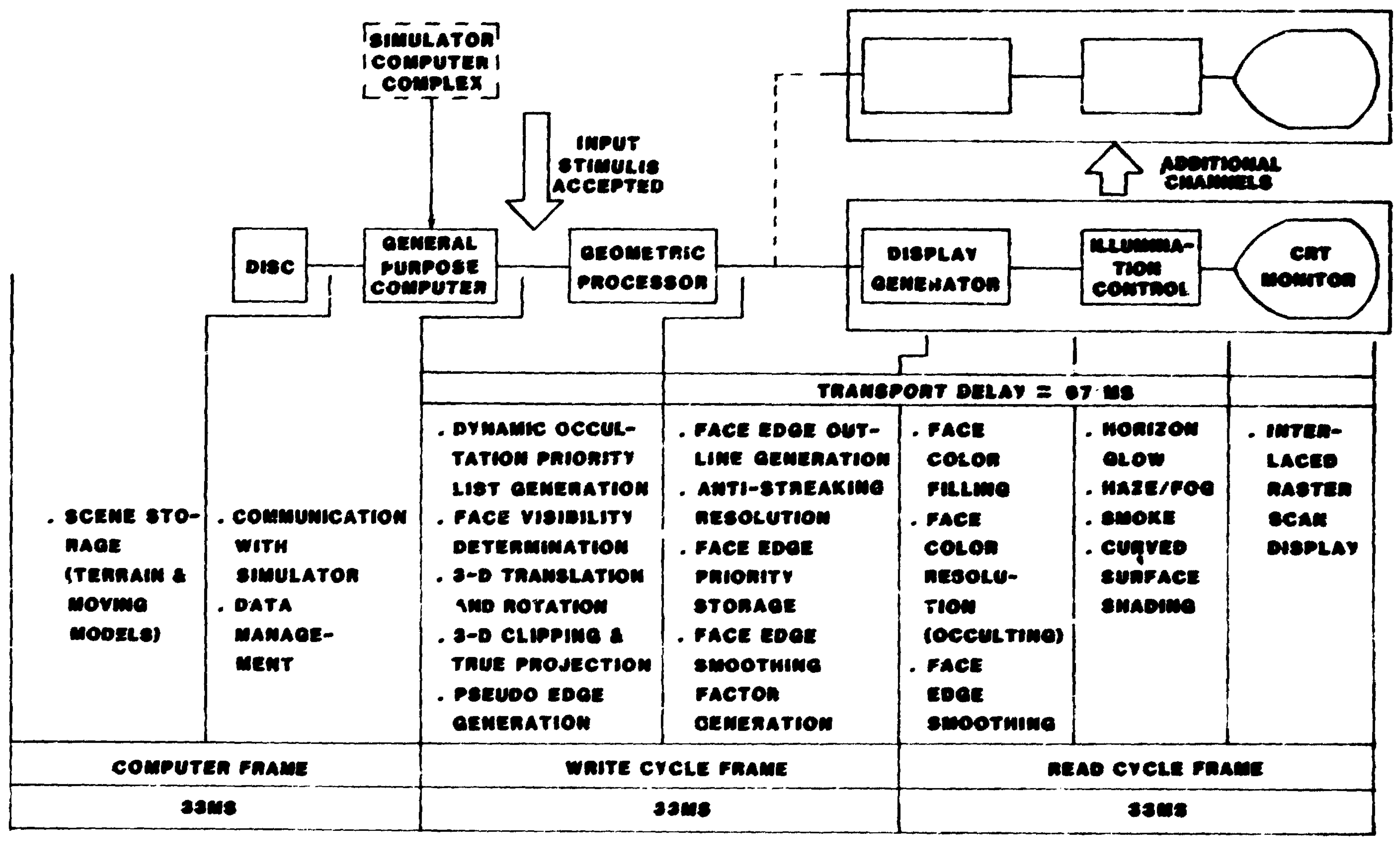

SIMULATOR COMPUTER COMPLEX
INPUT STIMULIS ACCEPTED
ADDITIONAL CHANNELS
DISC
GENERAL PURPOSE COMPUTER
GEOMETRIC PROCESSOR
DISPLAY GENERATOR
ILLUMINA-TION CONTROL
CRT MONITOR
TRANSPORT DELAY = 67 MS
. SCENE STO-RAGE (TERRAIN & MOVING MODELS)
. COMMUNICATION WITH SIMULATOR
. DATA MANAGE-MENT
. DYNAMIC OCCUL-TATION PRIORITY LIST GENERATION
. FACE VISIBILITY DETERMINATION
. 3-D TRANSLATION AND ROTATION
. 3-D CLIPPING & TRUE PROJECTION
. PSEUDO EDGE GENERATION
. FACE EDGE OUT-LINE GENERATION
. ANTI-STREAKING RESOLUTION
. FACE EDGE PRIORITY STORAGE
. FACE EDGE SMOOTHING FACTOR GENERATION
. FACE COLOR FILLING
. FACE COLOR RESOLU-TION (OCCULTING)
. FACE EDGE SMOOTHING
. HORIZON GLOW
. HAZE/FOG
. SMOKE
. CURVED SURFACE SHADING
. INTER-LACED RASTER SCAN DISPLAY
COMPUTER FRAME
33MS
WRITE CYCLE FRAME
33MS
READ CYCLE FRAME
33MS

Figure 4.6. GVS-1 Functional Allocation And Timing.

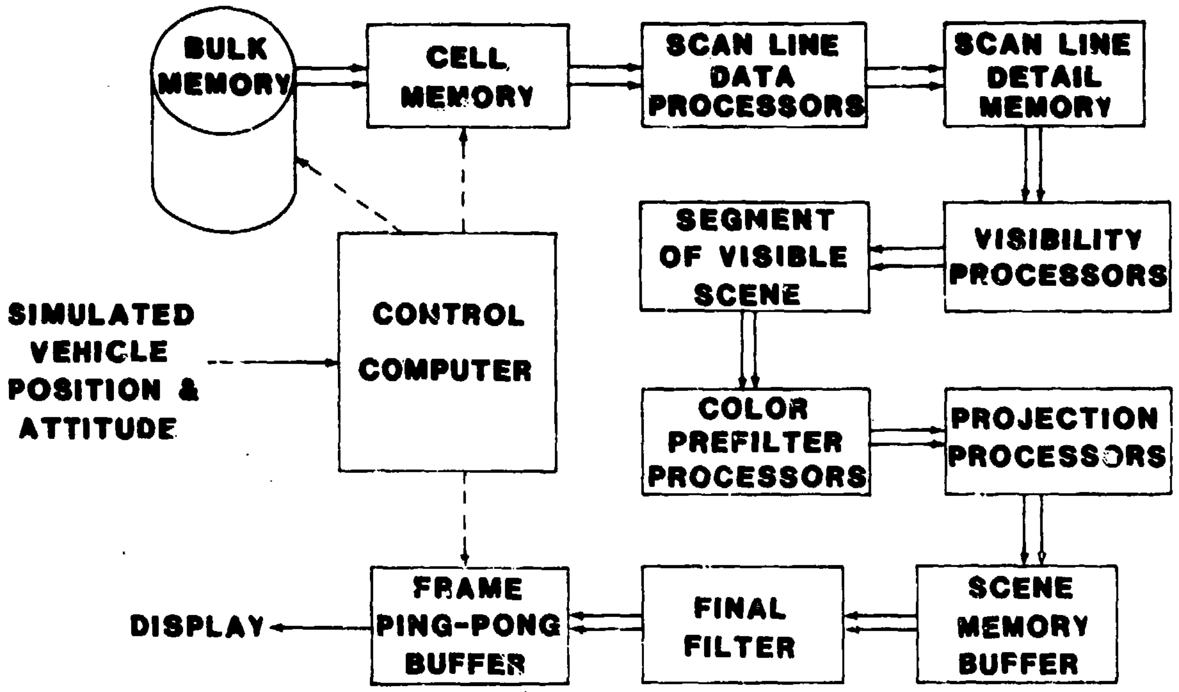


Figure 4.7. Functional Block Diagram of REALSCAN System.

REFERENCES

1. Computer Graphics, A Quarterly Report of SIGGRAPH-ACM, vol. 13, Aug. 1979.

2. W. K. Giloi, Interactive Computer Graphics. New Jersey: Prentice-Hall, pp. xi, 276-293, 1978.

3. G. V. Amico and C. E. Lindahl, "Real-Time Digital Simulation of Aircraft for Training Applications: Past, Present, and Future," Proc. Interservice/Industry Training Equip. Conf. and Exh. (Salt Lake City), Nov. 18-20, 1980.

4. R. A. Schumacker, "A New Visual System Architecture," Proc. I/ITEC, pp. 94-101, Nov. 1980.

5. W. Dichter, K. Doris, and C. Conkling, "A New Approach to CGI Systems," Proc. I/ITEC, pp. 102-109, Nov. 1980.

6. A. M. Spooner, D. R. Breglia, and B. W. Patz, "REALSCAN - A CIG System With Greatly Increased Image Detail," Proc. I/ITEC, pp. 110-116, Nov. 1980.

7. T. B. Cunningham and G. O. Picasso, "Automation of Data Base Development in Computer Image Generators," Proc. I/ITEC, pp. 17-21, Nov. 1980.

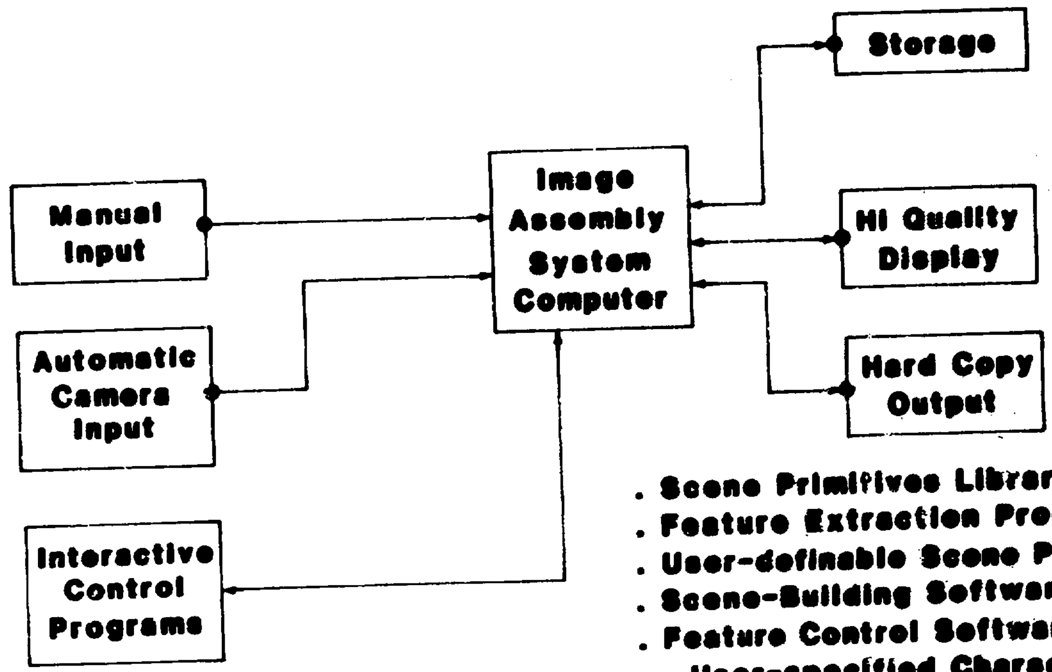
5. PERFORMANCE SPECIFICATIONS FOR THE IAS

The overall goal of this research was to specify an image assembly system that would provide a flexible and powerful tool for performing image understanding research.

A block diagram of the desired system is shown in Figure 5.1 The system had to be capable of both automatic camera input as well as manual input of two or three dimensional information. Both high quality real time image display and hard copy output were included. The image assembly system computer required both high speed and large storage capacity to permit processing of high resolution images in a reasonable time. Furthermore, special purpose hardware must be used to increase the speed of certain algorithms. The interactive control programs provide both image data base management and the processing power and flexibility required of a research facility.

The image assembly system software had to contain several special features, including a scene primitives library, a feature extraction programs library, the ability for user definable scene primitives, scene building software, feature control software with user-specified characteristics, and interactive re-definability for iterative processing. At present, no turn key system with the required characteristics has been found; however, it appears that the system could be assembled from existing components, with software development.

The results of the previous research and analysis are presented in this section as a performance specification for the Image Assembly System. The hardware specifications are described in Section 5.1, the software specifications in Section 5.2, and an example of an overall hardware system which could satisfy the requirements is described in Section 5.3.



. Scene Primitives Library
. Feature Extraction Programs Library
. User-definable Scene Primitives
. Scene-Building Software
. Feature Control Software
  User-specified Characteristics & Limits
. Interactive Redefinability
  (With Ease & Reasonable Speed)

Figure 5.1.

5.1. Hardware System

The overall Image Assembly Hardware System may be divided into two parts--general purpose computer hardware, and special purpose image and scene processing hardware. This division is necessary because no single turn key system with all the desirable features is currently availailable. The important features will now be summarized. These elements may also be considered as a distributed processing system with high speed local processing as required.

Some of the special purpose hardware features are shown below.

* Multiple image buffer display
* Transform processors for FFT, FHT, etc.
* Geometric transformation processor
* Hidden surface elimination
* Color function memories
* Polygonal area filling
* Windowing onto image planes
* High quality alphanumeric text system
* Vector drawing capability
* Control function menu
* Graphics overlay memories
* Vector convolution filters
* Contour tracing of boundaries

A comparison of the hardware components for the analysis and synthesis systems are shown in Table 5.1. Each hardware component listed is an element of either an image analysis or scene synthesis

	COMPONENT	ANALYSIS	SYNTHESIS	IAS
1.	General Computer with disks, tapes, etc.	Required	Required	Required
2.	Image Display with Multiple Memories	Required	Required	Required
3.	Low Resolution Image Digitizer	Required	Useful	Required
4.	High Resolution Image Digitizer	Required	Useful	Required
5.	Analog Video Recorder	Useful	Useful	Required
6.	Digitizing Tablet	Useful	Required	Required
7.	Vector Graphics Display	Useful	Required	Required
8.	Hidden Surface Hardware	Useful	Required	Required
9.	Matrix Multiplier	Useful	Required	Required
10.	Interactive Control	Required	Required	Required
11.	Image Transform Processor	Useful	Not Required	Required
12.	Floating Point Array Processor	Useful	Useful	Required

Table 5.1. Hardware Components.

system and is therefore a necessary element of the Image Assembly System (IAS). Each component and its required application will now be briefly described.

5.1.1 General Purpose Computer

Even though the IAS may be considered a distributed processing system, a central general purpose computer is required for both image analysis and scene synthesis. The parameters of this facility are dependent upon the number of users, the amount of distributed processing, and, in general, the larger the better. However, it is doubtful that even the world's most powerful computer, such as a CRAY, could perform the functions required at the speeds required for the IAS. For example, in image analysis, a 512 by 512 pixel two dimensional Fourier transform is a desirable operation which can be performed faster by an array processor than by the CRAY. In scene synthesis hidden surface elimination of a complex scene in 1/30th of a second is a desirable operation which can be performed faster in special purpose hardware. Both operations have wide application. The Fourier transform is useful in image enhancement, restoration, and compression. Hidden surface elimination is essential in any interactive scene display such as in flight simulation.

5.1.2 Image Display Device with Multiple Memories

A digital image display is essential in both image analysis and scene synthesis. Minimum display resolution for image processing is 512 by 512 pixels with three buffer memories storing 8 bits per pixel to provide full color capability. However, state of the art systems have capabilities which exceed this minimum requirement by a phenomenal amount. Available features include:

* As many as 16 refresh buffer memories, permitting coincident display of multispectral or derived images;
* Buffer memories larger than the display area. For example, a buffer memory of size 1024 by 1024 with a display size of 512 by 512 permits a 2 to 1 interactive roam of the displayed image throughout the stored image in both x and y directions;
* Direct input of digitized video into the buffer memories at TV rates;
* Function or transform memories with feedback loops which, when coupled with binary logarithms and antilogarithms, can add, subtract, multiply, divide, and compute arbitrary functions on the digital image data at TV rates;
* Hardware interpolation which permits interactive magnification or minification of an image. This feature is often mislabeled "zoom." A true zoom capability requires a perspective transformation--not just an affine transformation.
* Alphanumeric character generation;
* Overlay graphics buffer memories;
* Function plotting on the image raster;
* Hardware convolution with a small window function, thus permitting high speed operations such as edge detection

or noise smoothing by spatial filtering; and

* Interactive display control with menu selection, permitting ease of implementation of the display features.

These advances in image display systems in the last few years must be considered remarkable and attest both to the creativity of the designers and the decreasing memory and microprocessor costs.

Even with such tremendous advances in image display technology, operations which cannot be performed by the display alone are easily mentioned, such as the full size FFT or the hidden surface elimination. Therefore, other special purpose hardware is still required.

5.1.3 Low Resolution Image Digitizer

A standard TV resolution image digitizer is one of the most useful components in an image processing system, since it serves as an interface between the physical world and the computer. A well-designed digitization system with special purpose hardware controls can make the digitization of a new image easier than reading an image from digital tape.

Two special needs which may arise cannot be easily accommodated by most systems. The first is the ability to digitize video tape information without severe degradation. The second is the ability to digitize stereo cameras. Both problems are solvable; for example, a system recently constructed for NASA by the University of Tennessee has these and additional enhancement capabilities.

A low resolution digitizer might be characterized by a 500 point digitization over a one inch photocathode or a resolution of approximately 50 microns. For some applications, solid state camera sensors provide an adequate low resolution image.

5.1.4 High Resolution Image Digitizer

A high resolution image digitizer with resolution in the range of 1 to 10 microns is required for certain applications such as archivial digitization of source imagery. Both spatial and intensity linearity are crucial in these instruments.

5.1.5 Analog Video Recorder/Hard Copy

A color video recorder would be useful in both image analysis and scene synthesis to record time sequences of important phenomena. A hard copy unit with capabilities for polaroid photos or 35 mm slides is also useful. Movie equipment would also be useful.

5.1.6 Digitizing Tablets

One or more graphics digitizing tablets would be useful for certain scene synthesis operations. These tablets may be used for both control menu functions and for spatial position digitization.

5.1.7 Vector Graphics Display

A vector graphics display of much higher resolution than the image display would be useful in image analysis and would be required for scene synthesis.

5.1.8 Hidden Surface Hardware

Special purpose hardware is required for performing hidden surface elimination at TV rates for any but the most simple scenes. As described in the scene synthesis section, hidden line or surface elimination may be considered as a sorting problem. The conceptually simplest algorithm, the z buffer sort, simply fills the image buffer with the furthest object first and lets the nearer objects overwrite the further objects.

The hidden surface computation is the most difficult task in scene synthesis. However, several alternative implementations are now available from, for example, General Electric, Singer-Link, and Evans and Sutherland Corporations. Scene complexity is one of the limitations in all possible implementations; therefore, a careful match between system requirements and hardware capabilities is required.

5.1.9 Matrix Multiplier

A 4 by 4 matrix multiplication is required for scene transformations such as those required for translations, rotations, scale changes, and perspective changes. To accomplish these transformations for a scene data base or a high resolution image requires a computation time on the order of 100 nanoseconds for each displayed picture element. Therefore, a hardware matrix multiplier is required to accomplish these transformations at TV rates.

5.1.10 Interactive Control

Interactive control using such devices as the graphics tablet, joystick, trackball, light pen, mouse, or control function menu is now a practical alternative to the standard terminal keyboard. The use of interactive control provides an order of magnitude increase in ease of use, although some training time on each device is required. The most practical locations in the IAS for interactive control are in the image input and display stages, and most commercial devices have some of these features.

A simple cost analysis for each application should be done to show that the cost of the interactive control feature would be less than the labor cost to implement the task without the feature. The use of interactive control for the general image analysis and synthesis processing tasks is more difficult, since the tasks to be accomplished are not so well-defined. However, the use of variable function menus could permit easy implementation of most of the algorithms described in this report.

5.1.11 Image Transform Processor

An image transform processor which could, for example, compute a two dimensional Fourier transform at or near TV rates, would be useful in scene analysis for such operations as image enhancement or coding, and could also be used for scene synthesis operations such as texture synthesis. Since the fast algorithm implementations could be used, the number of operations required would be on the order of $2N^2\log_2 N$. For $N = 512$, this value is 4,718,592. To accomplish these operations in 1/30th of a second would permit only 7 nanoseconds/operation. For the fast Fourier transform, each operation is a complex add and multiply. Thus, it is seen that this requirement is very challenging, even with state of the art hardware. Special purpose hardware architecture is therefore required for these implementations. Some of the features that are used to accomplish the 7 nanoseconds/operation might include the following: simpler transforms could be used, such as the discrete cosine transform which requires only real operations; the Walsh/Hadamard transform, which requires only additions and subtractions; or the number theories which require only modulo arithmetic. An order of magnitude might be achieved this way, giving an operation time on the order of 70 nanoseconds/operation. Parallel computations implemented with an array processor could also be used. If 10 parallel operations are accomplished per cycle, then the operation time would be on the order of 700 nanoseconds per operation, which is well within the state of the art. Perhaps the highest degree of parallelism would be achieved with an optical implementation, which could compute the two dimensional Fourier transform in a time on the order of 1 nanosecond. Special architectures which use the image memories of the display and bit slice microprocessor technology are also now being used. The advent of large and very large scale integrated circuits (VLSI) should also provide new architectures for transform processing.

5.1.12 Floating Point Array Processor

A floating point array processor that could accomplish several floating point operations in parallel would be useful not only for image transforms, but also for any general computation requiring the use of floating point arithmetic. Such devices could be expected to provide a speed increase from 10 to 100 times the basic computer operation time. Because of the general nature of these devices, they would provide a means of adaption of the system capabilities to new algorithms.

6. CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

The basic objectives of this study were to investigate modern image analysis and scene synthesis techniques, including feature extraction processes in relation to the key components of a computer image generation, storage, and usage system. The main result of this effort, in addition to the summary of important image analysis and scene synthesis algorithms, is the design of an image assembly system (IAS) capable of both image analysis and synthesis which could provide a support research and development facility for advanced Air Force applications such as the generation of airborne sensor image predictions, flight simulator imagery, terminal guidance reference scene generation, and correlation algorithm evaluation, and studies of image compression and reconstruction.

The following engineering tasks have been accomplished.

1. A literature review covering image analysis, image processing, image understanding, scene synthesis, artificial intelligence, pattern recognition, and related subjects was performed to identify available technology applicable to the design of a computer-based image structuring system. The results of this review are discussed in Sections 2 and 3, and formed the basis of the design presented in Sections 4 and 5.

2. Research was performed to identify algorithms and techniques for effectively representing image and scene information. These algorithms are described in Sections 2 and 3.

3. An Image Assembly System (IAS) design was constructed in terms of major general-purpose and special-purpose hardware elements, and is presented in terms of functional hardware and software requirements in Section 5.

6.2 Comments

The basic idea presented in this report is the design of an Image Assembly System capable of both image analysis and scene synthesis. This is a novel concept, and may be considered by some as more futuristic than realistic. This argument will be most strongly presented by those familiar with either the image analysis or scene synthesis areas, but not those familiar with both. Therefore, some of the overall concepts will now be reviewed, along with a discussion of considerations leading to a recommendation for further research.

First, in all design disciplines, analysis and synthesis are closely related concepts. In electronics, for example, students are first taught to analyze circuits as the first step in the synthesis of new circuits. Similar examples can be drawn from all design disciplines in engineering, architecture, and science.

Perhaps the fundamental reason is that the design parameters which must be specified in synthesis are understood only through analysis.

Secondly, most areas of interest in advanced Air Force research are so difficult that only limited solutions are available in the current state of the art. An all weather, all capability, weapons or intelligence system in a single device is probably not possible. In any single application area involving images or scenes, the fundamental scene complexity problem will be encountered. The fact that there is not enough storage capability in the human brain (10^{14} bits) to store the locations of atoms in a microgram of salt (10^{16} bits) is a clear illustration of the scene complexity limitation. Even with the recognition of this limitation, very effective and accurate systems have and can be developed. These systems rely on the characterization of key important scene features. Therefore, the extraction of important features through analysis and the presentation of these features through synthesis is the basis of the disciplines called image analysis, image processing, scene synthesis, computer graphics, artificial intelligence, pattern recognition, and computer vision. Certainly the combination of these distinct areas is difficult; however, a research facility that could accomodate most areas should permit the study of not only each area alone, but in a rich variety of combinations as well.

One question which might arise is if there are personnel who are t[illegible]ed in both image analysis and synthesis. One indication that the answer is positive is the cooperation between the IEEE

Computer Society members interested in image analysis and the ACM Special Interest Group on Computer Graphics. The annual meetings of these two groups have been held at the same time and place for the past two years, and conjunctive meetings have been projected for 1981 and 1982. Thus, on the national and international level, there is an indication of common interest. Another indicator is the journal, Computer Graphics and Image Processing, edited by Professor Azriel Rosenfeld. A final indicator on the local level are the Computer Graphics and Image Processing courses offered in the Departments of Electrical Engineering and Computer Science at the University of Tennessee. Other universities, such as Maryland and UCLA, also offer this type training. Admittedly, it will be harder to find personnel trained in both areas than those trained in one or the other separately; however, the number of personnel required for research is relatively small.

In terms of computer hardware requirements, there is more hardware required for the combined capability of analysis and synthesis than that required for either separately. However, there is a great deal of common hardware, as illustrated in Section 5. In fact, almost every hardware element required for one system is useful in extending the capabilities of the other system. A similar arguments could be made for the software requirements.

The final arguments for the combined IAS capability is perhaps the realization that advances beyond the state of the art will require both capabilities. Realistic scenes will perhaps never be generated until enough real images are analyzed to permit the identification of all key features required for the synthesis. Similarly, reliable image analysis algorithms may never be developed until it is possible to test these algorithms on a great variety of realistic synthesized images.

6.3 Recommendations for Further Research

Although significant progress has been made during this effort toward the specification of methodology and algorithms for the efficient analysis and synthesis of scene data for advanced Air Force applications, further research in several areas is indicated in order to further define the problems, refine the algorithms described, study the hardware and software tradeoffs, and develop required techniques. These areas will now be outlined.

Research should be conducted to identify algorithms and techniques fc fficiently representing image and scene information in digital form. The representation of both planar and curved surfaces in a data base design suited to scene synthesis algorithms such as hidden surface elimination as well as other scene reflectance transformations is a key area for further research. The use of this data structure for multi-sensor surface reflectance computations, high speed hidden surface display algorithms, and real time scene transformations should be considered. Provisions for automatic and interactive input of scene data in an efficient manner, such as a hierarchical design, should also be considered. This work would be especially applicable to reference scene preparation for terminal guidance.

Research should also be conducted to develop methods for image analysis and image data base designs. Algorithms for the extraction of scene features through regional and relational segmentation, texture recognition and measurement, object detection, recognition and location from multi-spectral image data bases should be studied. Important problem areas and design tradeoffs for real time analysis should also be examined. Digital scene representations using hierarchical structural forms for algorithm efficiency should be considered. Finally, provisions for both automatic and interactive extraction of image and scene information should be studied.

Research to further refine the capabilities and requirements of a combined scene synthesis and analysis system should also be conducted. This unique combination promises to provide the most flexible and powerful image research facility ever assembled and would provide an important tool for the solution of advanced Air Force applications.

Acknowledgements

The authors would like to thank Dr. M. G. Thomason and Dr. J. J. Hwang for many helpful discussions and for their contributions to this report.